



**Devang Patel Institute of
Advance Technology and Research**
(A Constitute Institute of CHARUSAT)

Certificate

This is to certify that

Mr./Mrs. Prabin Bhagchandani

of Computer Engineering Class,

ID. No. 22 DCE006 has satisfactorily completed

his/ her term work in CE361 - Operating System for

the ending in Nov 2024/2025

Date :

Sign. of Faculty

Head of Department



Practical 1

Aim: Introduction to OS and shell.

1. Access the command Line
2. Manage files and directories from command line
3. Create, edit and view text files

Commands for reference:

Directory: mkdir, rmdir, cd, pwd, ls, mv

Editor: vi, gedit

File Handling/Text: cp, mv, rm, sort, cat, file, less, more, cmp, diff, comm, head, tail, cut, grep, touch, tr, uniq

Self-Study:

User Access: login, logout, passwd, exit

Information: man, who, date, cal, tty, calendar, time, bc, who, whoami, which, hostname, history, wc, finger, uname

Help: man, help

Terminal: echo, clear

Exercise - 0

Enter these commands at the UNIX prompt, and try to interpret the output.

Que.	1. Passwd
Command	1.passwd
Output	<code>ubuntu@ubuntu:~\$ passwd</code> New password:

Que.	2. Date 3. Hostname 4. Arch 5. uname –a
Command	2. date

	3. hostname 4. arch 5. uname -a	
Output	<pre>ubuntu@ubuntu:~\$ date Thu Jul 11 14:39:43 IST 2024 ubuntu@ubuntu:~\$ hostname ubuntu ubuntu@ubuntu:~\$ arch x86_64 ubuntu@ubuntu:~\$ uname -a Linux ubuntu 6.8.0-31-generic #31-Ubuntu SMP PREEMPT_DYNAMIC Sat Apr 20 00:40:06 UTC 2024 x86_64 x86_64 x86_64 GNU/Linux</pre>	

Que.	6. whoami 7. who 8. id 9. echo \$SHELL 10. echo {con,pre} {sent,fer} {s,ed}
Command	6. whoami 7. who 8. id 9. echo \$SHELL 10. echo {con,pre} {sent,fer} {s,ed}
Output	<pre>ubuntu@ubuntu:~\$ whoami ubuntu ubuntu@ubuntu:~\$ who ubuntu seat0 2024-06-27 15:41 (login screen) ubuntu :0 2024-06-27 15:41 (:0) ubuntu@ubuntu:~\$ id uid=1000(ubuntu) gid=1000(ubuntu) groups=1000(ubuntu),4(adm),24(cdrom),27(sudo), 30(dip),46(plugdev),100(users),114(lpadmin),124(sambashare) ubuntu@ubuntu:~\$ echo \$SHELL /bin/bash ubuntu@ubuntu:~\$ echo {con,pre}{sent,fer}{s,ed} consents consented confers conferred presents presented prefers prefered</pre>

Que.	11. man ls 12. man who
Command	11. man ls 12. man who

Output

```
LS(1)                               User Commands                               LS(1)

NAME
    ls - list directory contents

SYNOPSIS
    ls [OPTION]... [FILE]...

DESCRIPTION
    List information about the FILEs (the current directory by default).
    Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

    Mandatory arguments to long options are mandatory for short options too.

    -a, --all
        do not ignore entries starting with .

    -A, --almost-all
        do not list implied . and ..

    --author
Manual page ls(1) line 1 (press h for help or q to quit)
```

```
WHO(1)                               User Commands                               WHO(1)

NAME
    who - show who is logged on

SYNOPSIS
    who [OPTION]... [FILE | ARG1 ARG2]

DESCRIPTION
    Print information about users who are currently logged in.

    -a, --all
        same as -b -d --login -p -r -t -T -u

    -b, --boot
        time of last system boot

    -d, --dead
        print dead processes

    -H, --heading
        print line of column headings

Manual page who(1) line 1 (press h for help or q to quit)
```

Que.	13. who can tell me why I got divorced 14. clear
Command	13. who can tell me why I got divorced 14. clear
Output	ubuntu@ubuntu:~\$ who can tell me why i got divorced who: extra operand ‘me’ Try ‘who --help’ for more information.

Que.	15. cal 2000
Command	15. sudo apt install ncal

cal 2000

Output

```
ubuntu@ubuntu:~$ cal 2000
          2000
January           February          March
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
      1       1  2  3  4  5       1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
 2  3  4  5  6  7  8   6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
 9 10 11 12 13 14 15 13 14 15 16 17 18 19 20 21 22 23 24 25 26 19 20 21 22 23 24 25
16 17 18 19 20 21 22 20 21 22 23 24 25 26 19 20 21 22 23 24 25
23 24 25 26 27 28 29 27 28 29 26 27 28 29 30 31
30 31

April            May             June
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
      1       1  2  3  4  5  6       1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17
 2  3  4  5  6  7  8   7  8  9 10 11 12 13 14 15 16 17 18 19 20 11 12 13 14 15 16 17
 9 10 11 12 13 14 15 14 15 16 17 18 19 20 16 17 18 19 20 21 22 23 24 25 26 27 18 19 20 21 22 23 24
16 17 18 19 20 21 22 21 22 23 24 25 26 27 23 24 25 26 27 28 29 30 31 25 26 27 28 29 30
30

July            August          September
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
      1       1  2  3  4  5  6       1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17
 2  3  4  5  6  7  8   6  7  8  9 10 11 12 13 14 15 16 17 18 19 10 11 12 13 14 15 16
 9 10 11 12 13 14 15 13 14 15 16 17 18 19 20 16 17 18 19 20 21 22 23 24 25 26 17 18 19 20 21 22 23
16 17 18 19 20 21 22 20 21 22 23 24 25 26 23 24 25 26 27 28 29 30 31 24 25 26 27 28 29 30
30 31

October         November        December
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
      1       1  2  3  4  5  6       1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17
 1  2  3  4  5  6  7   5  6  7  8  9 10 11 12 13 14 15 16 17 18 10 11 12 13 14 15 16
 8  9 10 11 12 13 14 12 13 14 15 16 17 18 19 15 16 17 18 19 20 21 22 23 24 25 17 18 19 20 21 22 23
15 16 17 18 19 20 21 19 20 21 22 23 24 25 22 23 24 25 26 27 28 29 30 24 25 26 27 28 29 30
29 30 31 26 27 28 29 30 31
```

Que.

16. cal 9 1752
17. bc -l
18. echo 5+4 | bc -l

Command

16. cal 9 1752
17. bc -l
18. echo 5+4 | bc -l

Output

```
ubuntu@ubuntu:~$ cal 9 1752
September 1752
Su Mo Tu We Th Fr Sa
      1  2 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30

ubuntu@ubuntu:~$ bc -l
bc 1.07.1
Copyright 1991-1994, 1997, 1998, 2000, 2004, 2006, 2008, 2012-2017 Free Software
Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type `warranty'.

ubuntu@ubuntu:~$ echo 5+4 | bc -l
9
```

Que.	19. yes please 20. time sleep 5
Command	19. yes please 20. time sleep 5
Output	<pre>please please please please please please please please please pleas^C ubuntu@ubuntu:~\$ time sleep 5 real 0m6.649s user 0m0.008s sys 0m0.008s</pre>

Que.	21. history
Command	21. history
Output	<pre>ubuntu@ubuntu:~\$ history 1 date 2 cd 3 date 4 passwd 5 date 6 hostname 7 arch 8 uname-a 9 whoami 10 who 11 id 12 echo \$SHELL 13 echo {con,pre}{sent,fer}{s,ed} 14 man ls 15 man who 16 who can tell me why i got divorced 17 cal 2000 18 cal 2024 19 cal 9 1752 20 bc -l 21 bc -l 22 echo 5+4 bc -l 23 yes please</pre>

Exercise-1

Try the following command sequence.

Que.	1. Display username of current user. 2. Display current working directory. 3. Make a sub directory named CE in a directory named CSPIT. 4. Create an empty file “ce1.txt” from command prompt.
Command	1. whoami 2. pwd 3. mkdir -p CSPIT/CE 4. touch CSPIT/CE/ce1.txt
Output	<pre>ubuntu@ubuntu:~\$ whoami ubuntu ubuntu@ubuntu:~\$ pwd /home/ubuntu ubuntu@ubuntu:~\$ mkdir -p CSPIT/CE ubuntu@ubuntu:~\$ touch CSPIT/CE/ce1.txt ubuntu@ubuntu:~\$</pre>

Que.	5. Add the content from command prompt in “ce1.txt”. 6. Display the content of “ce1.txt” file. 7. Change working directory to CE. 8. Make 5 empty files named file1.txt to file5.txt in same directory. 9. List all the files in the directory CE. 10. Add the Name, ID no, and address with pin code to “file1.txt”. 11. Copy contents of file1.txt to file2.txt. 12. Rename file3.txt to “f3.txt”. 13. Display the number of lines, number of words, number of characters of “file1.txt”.
Command	5. echo -e “Probin” 6. cat file1.txt 7. cd CSPIT/CE 8. touch file{1..5}.txt 9. ls 10. echo -e “Name: Probin \nID: 22DCE006\nAddress: Changa”>file1.txt 11. cp file1.txt file2.txt

12. mv file3.txt f3.txt
 13. wc file1.txt

Output

```
ubuntu@ubuntu:~$ cd CSPIT/CE
ubuntu@ubuntu:~/CSPIT/CE$ touch file{1..5}.txt
ubuntu@ubuntu:~/CSPIT/CE$ ls
ce1.txt file1.txt file2.txt file3.txt file4.txt file5.txt
ubuntu@ubuntu:~/CSPIT/CE$ echo -e "Name:Probin\nID:22DCE006\nAddress:Changa">file1.txt
ubuntu@ubuntu:~/CSPIT/CE$ cp file1.txt file2.txt
ubuntu@ubuntu:~/CSPIT/CE$ mv file3.txt f3.txt
ubuntu@ubuntu:~/CSPIT/CE$ wc file1.txt
 1 1 43 file1.txt
```

- | | |
|-------------|---|
| Que. | <p>14. Compare the files “file1.txt” to “file2.txt”
 15. Update the content of “file2.txt”. Add your skill to existing file.
 16. Compare the files “file1.txt” to “file2.txt”
 17. Report what is common in the above given files.</p> |
|-------------|---|

**Comman
d**

14. diff file1.txt file2.txt
 15. echo “Skills : JAVA” >> file2.txt
 16. diff file1.txt file2.txt
 17. comm -12 <(sort file1.txt) <(sort file2.txt)

Output

```
ubuntu@ubuntu:~/CSPIT/CE$ diff file1.txt file2.txt
ubuntu@ubuntu:~/CSPIT/CE$ echo "Skills:JAVA">>>file2.txt
ubuntu@ubuntu:~/CSPIT/CE$ diff file1.txt file2.txt
1a2
> Skills:JAVA
ubuntu@ubuntu:~/CSPIT/CE$ comm -12<(sort file1.txt)<(sort file2.txt)
' ' syntax error near unexpected token `newline'
op Center syntax error
ubuntu@ubuntu:~/CSPIT/CE$ comm -12<(sort file1.txt)<(sort file2.txt)
comm: invalid option -- '/'
Try 'comm --help' for more information.
ubuntu@ubuntu:~/CSPIT/CE$ comm -12<(sort file1.txt)<(sort file2.txt)
comm: invalid option -- '/'
Try 'comm --help' for more information.
ubuntu@ubuntu:~/CSPIT/CE$ comm -12<(sort file1.txt) <(sort file2.txt)
comm: invalid option -- '/'
Try 'comm --help' for more information.
ubuntu@ubuntu:~/CSPIT/CE$ comm -12 <(sort file1.txt) <(sort file2.txt)
-eName:Probin\nID:22DCE006\nAddress:Changa
ubuntu@ubuntu:~/CSPIT/CE$
```

- | | |
|-------------|--|
| Que. | <p>18. Add the content in “file4.txt” as given:
 India
 United States of America</p> |
|-------------|--|

	United Kingdom Australia
Command	18. cat > file4.txt India United States of America United Kingdom Australia
Output	<pre>ubuntu@ubuntu:~/CSPIT/CE\$ echo -e "India\nUnited States of America\nUnited Kingdom\nAustralia">file4.txt ubuntu@ubuntu:~/CSPIT/CE\$ cat CSPIT/CE/file4.txt cat: CSPIT/CE/file4.txt: No such file or directory ubuntu@ubuntu:~/CSPIT/CE\$ cat file4.txt India United States of America United Kingdom Australia ubuntu@ubuntu:~/CSPIT/CE\$</pre>

Que.	19. Add the content in “file5.txt” as given: India Canada United Kingdom Australia Germany 20. Find the difference between “file4.txt” and “file5.txt”. How to make these files identical?
Comma nd	19. cat > file5.txt India Canada United Kingdom Australia Germany 20. diff file4.txt file5.txt

Output	<pre> ubuntu@ubuntu:~/CSPIT/CE\$ echo -e "India > Canada > United Kingdom > Australia > Germany">file5.txt ubuntu@ubuntu:~/CSPIT/CE\$ cat file5.txt -eIndia Canada United Kingdom Terminal Germany ubuntu@ubuntu:~/CSPIT/CE\$ diff file4.txt file5.txt 1,2c1,2 < India < United States of America --- > -eIndia > Canada 4a5 > Germany ubuntu@ubuntu:~/CSPIT/CE\$ </pre>
---------------	--

Que.	<p>21. Create “file6.txt” by adding ten name of students. 22. Create “file7.txt” by adding ten name of students.(few names should be common to “file6.txt”)</p>
Comma nd	<p>21. echo -e “david raj prem josh raju shyam juhi hetvi aayush john ”>file6.txt cat file6.txt 22.echo -e “raj prem josh raju shyam ranjesh ranjan diya deepika ”>file7.txt cat file7.txt</p>

Output

```

> Germany
ubuntu@ubuntu:~/CSPIT/CE$ echo -e "david raj prem josh raju shyam juhi hetvi
ohn">file6.txt
ubuntu@ubuntu:~/CSPIT/CE$ cat file6.txt
Terminal :aj prem josh raju shyam juhi hetvi aayush john
ubuntu@ubuntu:~/CSPIT/CE$ echo -e "david raj prem josh raju shyam juhi karan
alu">file6.txt
ubuntu@ubuntu:~/CSPIT/CE$ cat file6.txt
-edavid raj prem josh raju shyam juhi karan nityam lalu
ubuntu@ubuntu:~/CSPIT/CE$ cat file6.txt
-edavid raj prem josh raju shyam juhi rohan ram shyam
ubuntu@ubuntu:~/CSPIT/CE$ echo -e "david raj prem josh raju shyam rajesh ranjan
deepkia">file7.txt
ubuntu@ubuntu:~/CSPIT/CE$ cat file7.txt
-edavid raj prem josh raju shyam rajesh ranjan diya deepkia
ubuntu@ubuntu:~/CSPIT/CE$ 

```

Que.

23. Sort the content of “file6.txt” and “file7.txt”
24. Find the common and unique content in “file6.txt” and “file7.txt”
25. Merge the content of above two files in “file8.txt”
26. Remove the duplicate names from “file8.txt”

Command

23. sort file6.txt -o file6.txt
sort file7.txt -o file7.txt
24. comm -12 file6.txt file7.txt
comm -23 file6.txt file7.txt
comm -13 file6.txt file7.txt
25. cat file6.txt file7.txt > file8.txt
26. sort -u file8.txt -o file8.txt
cat file8.txt

Output

```

ubuntu@ubuntu:~/CSPIT/CE$
ubuntu@ubuntu:~/CSPIT/CE$ sort file6.txt -o file6.txt
ubuntu@ubuntu:~/CSPIT/CE$ sort file7.txt -o file7.txt
ubuntu@ubuntu:~/CSPIT/CE$ comm -12 file6.txt file7.txt
ubuntu@ubuntu:~/CSPIT/CE$ comm -23 file6.txt file7.txt
-edavid raj prem josh raju shyam juhi rohan ram shyam
ubuntu@ubuntu:~/CSPIT/CE$ comm -13 file6.txt file7.txt
-edavid raj prem josh raju shyam rajesh ranjan diya deepkia
ubuntu@ubuntu:~/CSPIT/CE$ cat file6.txt file7.txt>file8.txt
ubuntu@ubuntu:~/CSPIT/CE$ sort -u file8.txt -o file8.txt
ubuntu@ubuntu:~/CSPIT/CE$ cat file8.txt
-edavid raj prem josh raju shyam juhi rohan ram shyam
-edavid raj prem josh raju shyam rajesh ranjan diya deepkia

```

Que.

27. Translate the content of “file1.txt”:
 - a. Lower case to upper case

	b. Remove digits from file
Comma nd	27. a. tr '[:lower:]' '[:upper:]' < file1.txt > file1_no_digits.txt b. tr -d '[:digit:]' < file1.txt > file1_no_digits.txt cat file1_no_digits.txt
Output	<pre>ubuntu@ubuntu:~/CSPIT/CE\$ ubuntu@ubuntu:~/CSPIT/CE\$ tr '[:lower:]' '[:upper:]' <file1.txt>file1_no_digits.txt ubuntu@ubuntu:~/CSPIT/CE\$ cat file1.txt -eName:Probin\nID:22DCE006\nAddress:Changa ubuntu@ubuntu:~/CSPIT/CE\$ tr -d '[:digit:]' <file1.txt>file1_no_digits.txt ubuntu@ubuntu:~/CSPIT/CE\$ cat file1_no_digits.txt -eName:Probin\nID:DCE\nAddress:Changa ubuntu@ubuntu:~/CSPIT/CE\$</pre>

Que.	28. Apply head and tail command to see the content of “file8.txt” with different arguments.
Command	28. head -n 5 file8.txt tail -n 5 file8.txt
Output	<pre>ubuntu@ubuntu:~/CSPIT/CE\$ head -n 5 file8.txt -edavid raj prem josh raju ubuntu@ubuntu:~/CSPIT/CE\$ tail -n 5 file8.txt shyam rajesh ranjan diya deepkia ubuntu@ubuntu:~/CSPIT/CE\$</pre>

Que.	29. Differentiate between less and more command and check why less is faster than more command. 30. Create a file “file9.txt” having content: Linux is great os. Linux is open source. Linux is free os. You can learn operating system with linux. Unix or linux which one you choose. liNux is easy to learn. Linux is a multiuser os. Learn linux. Linux is a powerful. 31. Find the lines which contains “linux”.
Comma nd	29. less file8.txt more file8.txt 30. cat > file9.txt Linux is great os. Linux is open source. Linux is free os.

You can learn operating system with linux.
 Unix or linux which one you choose.
 liNux is easy to learn. Linux is a multiuser os. Learn linux. Linux is a powerful.
 31. grep -i 'linux' file9.txt

Output	<pre>ubuntu@ubuntu:~/CSPIT/CE\$ cat file9.txt Linux is great os. Linux is open source. Linux is free os. You can learn operating system with linux. Unix or linux which one you choose. liNux is easy to learn. Linux is multiuser os. Learn linux. Linux is a ubuntu@ubuntu:~/CSPIT/CE\$ grep -i "linux" file9.txt Linux is great os. Linux is open source. Linux is free os. You can learn operating system with linux. 910 MB Volume Unix or Linux which one you choose. liNux is easy to learn. Linux is multiuser os. Learn linux. Linux is a ubuntu@ubuntu:~/CSPIT/CE\$</pre>
---------------	---

Que.	32. Count the number of lines that matches the “linux” 33. Show the line number of file with the line matched 34. Find the lines which start with “linux” 38. Check the file type of lab manual and other files created. 39. Apply history command and redirect your output to “ID No_date.txt”
Command	32. grep -i 'linux' file9.txt wc -l 33. grep -in 'linux' file9.txt 34. grep -i '^linux' file9.txt 38. file file9.txt File view 39. history > 22DCE006_\$(date +%Y%m%d).txt
Output	<pre>ubuntu@ubuntu:~/CSPIT/CE\$ grep -i "linux" file9.txt wc -l 4 ubuntu@ubuntu:~/CSPIT/CE\$ grep -in "linux" file9.txt 1:Linux is great os. Linux is open source. Linux is free os. 2:You can learn operating system with linux. 3:Unix or Linux which one you choose. 4:liNux is easy to learn. Linux is a multiuser os. Learn linux. Linux is a power ful. ubuntu@ubuntu:~/CSPIT/CE\$ grep -i "^linux" file9.txt Linux is great os. Linux is open source. Linux is free os. liNux is easy to learn. Linux is a multiuser os. Learn linux. Linux is a powerf l.</pre>

Que.	35. Find the lines which ends with “os”. 36. Display the file name that contains “linux”.
Command	35. grep -i 'os\$' file9.txt 36. grep -il 'linux' file9.txt

Output	<pre>ubuntu@ubuntu:~/CSPIT/CE\$ ubuntu@ubuntu:~/CSPIT/CE\$ grep -i "os\$" file9.txt ubuntu@ubuntu:~/CSPIT/CE\$ grep -il "linux" file9.txt file9.txt ubuntu@ubuntu:~/CSPIT/CE\$</pre>
---------------	---

Que.	37. Download lab manual from department course website.
Command	37. wget <URL>
Output	<pre>ubuntu@ubuntu:~\$ wget https://drive.google.com/file/d/14ru3X0tNv_dhyWQjeB5061p02wFESKP6/view --2024-07-11 09:23:58-- https://drive.google.com/file/d/14ru3X0tNv_dhyWQjeB5061p02wFESKP6/view Resolving drive.google.com (drive.google.com)... 142.250.183.78, 2404:6800:4009:814::200e Connecting to drive.google.com (drive.google.com) 142.250.183.78 :443... connected. HTTP request sent, awaiting response... 200 OK Length: unspecified [text/html] Saving to: 'view' view [=>] 90.59K 315KB/s in 0.3s 2024-07-11 09:24:00 (315 KB/s) - 'view' saved [92764]</pre>

Exercise - 1.1 (Advanced)

Try the following command sequence.

Que.	<ol style="list-style-type: none">1. Change back into your home directory.2. Make subdirectories called work and play3. Delete the subdirectory called work.4. Copy the file /etc/passwd into your home directory.5. Move it into the subdirectory play6. What is the difference between listing the contents of directory play with ls -l and ls -L?7. Create a file called hello.txt that contains the words "hello world". Can you use "cp" using "terminal" as the source file to achieve the same effect?
Command	<ol style="list-style-type: none">1. cd ~2. mkdir work play3. rmdir work4. cp /etc/passwd ~5. mv ~/passwd ~/play/6. ls -l (Lists files with detailed information including permissions, number of links, owner, group, size, and timestamp.) Ls -L {follows symbolic links (shows the file or directory the link points to)}7. cat > hello.txt Hello world or echo "hello world" cp /dev/stdin hello.txt

Output

```
ubuntu@ubuntu:~/CSPIT/CE$ cd ~
ubuntu@ubuntu:~$ mkdir work play
ubuntu@ubuntu:~$ rmdir work
ubuntu@ubuntu:~$ cp /etc/passwd ~
ubuntu@ubuntu:~$ mv ~/passwd ~/play/
ubuntu@ubuntu:~$ ls -l
total 0
drwxrwxr-x 3 ubuntu ubuntu 60 Jul  7 04:16 CSPIT
drwxr-xr-x 2 ubuntu ubuntu 60 Jul  7 03:39 Desktop
drwxr-xr-x 2 ubuntu ubuntu 40 Jul  7 03:40 Documents
drwxr-xr-x 2 ubuntu ubuntu 40 Jul  7 03:40 Downloads
drwxr-xr-x 2 ubuntu ubuntu 40 Jul  7 03:40 Music
drwxr-xr-x 2 ubuntu ubuntu 40 Jul  7 03:40 Pictures
drwxr-xr-x 2 ubuntu ubuntu 40 Jul  7 03:40 Public
drwxr-xr-x 2 ubuntu ubuntu 40 Jul  7 03:40 Templates
drwxr-xr-x 2 ubuntu ubuntu 40 Jul  7 03:40 Videos
drwxrwxr-x 2 ubuntu ubuntu 60 Jul  7 10:37 play
drwx----- 6 ubuntu ubuntu 120 Jul  7 06:09 snap
ubuntu@ubuntu:~$ ls -L
CSPIT  Documents  Music  Public  Videos  snap
Desktop  Downloads  Pictures  Templates  play
ubuntu@ubuntu:~$ cat > hello.txt
hello world
```

Que.

8. Copy hello.txt to terminal. What happens?
9. Imagine you were working on a system and someone accidentally deleted the ls command (/bin/ls). How could you get a list of the files in the current directory? Try it. (Do not delete ls command, copy it to some other location from bin)
10. How would you create and then delete a file called "\$SHELL"? Try it.
11. How would you create and then delete a file that begins with the symbol #? Try it
12. How would you create and then delete a file that begins with the symbol -? Try it.

Command

8. cp hello.txt terminal (As terminal named file is not present, therefore file with name 'terminal' gets created and content of file hello.txt copied to it.)
9. echo *
10. touch '\$SHELL'
rm '\$SHELL'
11. touch '#new_file'
rm '#new_file'
12. touch - '-file'
rm - '-file'

Output

```
ubuntu@ubuntu:~$ cp hello.txt terminal
ubuntu@ubuntu:~$ echo *
CSPIT Desktop Documents Downloads Music Pictures Public Templates Videos hello.txt p
lay snap terminal
ubuntu@ubuntu:~$ touch '$SHELL'
ubuntu@ubuntu:~$ rm '$SHELL'
ubuntu@ubuntu:~$ touch '#new_file'
ubuntu@ubuntu:~$ rm '#new_file'
ubuntu@ubuntu:~$ touch -- '-file'
ubuntu@ubuntu:~$ rm -- '-file'
```

Practical 2

Aim: User Administration

1. Manage local users, groups and creation of multiple users from excel sheet

2. Control access to files

Commands for reference:

System Administrator: su, adduser, addgroup, rmuser, shutdown

Control Access: chmod, umask

PART A

Manage local users, groups and creation of multiple users from excel sheet

Que.	<ol style="list-style-type: none"> Run id command to view the current user and group information. display the current working directory. print the value of HOME and PATH variable to determine the home directory and user's executable's path respectively. Run su and su - command. Observe the output for the same.what is the main difference between them? Run sudo su at the shell prompt to become the root user. Exit the current user's shell to return to the student user's shell
Command	<ol style="list-style-type: none"> id pwd echo \$HOME echo \$PATH su su - <ul style="list-style-type: none"> • su: Switches to the target user's shell but retains the current environment. • su -: Switches to the target user's shell and initiates a new login shell, effectively switching to the target user's environment. sudo su exit

Output	<pre> ubuntu@ubuntu:~\$ id uid=1000(ubuntu) gid=1000(ubuntu) groups=1000(ubuntu),4(adm),24(cdrom),27(sudo), 30(dip),46(plugdev),100(users),114(lpadmin),124(sambashare) ubuntu@ubuntu:~\$ pwd /home/ubuntu ubuntu@ubuntu:~\$ echo \$HOME /home/ubuntu ubuntu@ubuntu:~\$ echo \$PATH /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/snap/bin ubuntu@ubuntu:~\$ su Password: ubuntu@ubuntu:~\$ sudo su root@ubuntu:/home/ubuntu# ^C root@ubuntu:/home/ubuntu# ^C root@ubuntu:/home/ubuntu# exit exit ubuntu@ubuntu:~\$</pre>
Que	10. Attempt to view the last five lines of /var/log/auth.log without using sudo 11. Attempt to view the last five lines of /var/log/auth.log using sudo 12. Attempt to make a copy of /etc/rpc as /etc/rpcOLD without using sudo 13. Attempt to make a copy of /etc/rpc as /etc/rpcOLD with sudo. 14. Attempt to delete /etc/rpcOLD without using sudo 15. Attempt to delete /etc/rpcdOLD with sudo
Commands	10.tail -n 5 /var/log/auth.log 11.sudo tail -n 5 /var/log/auth.log 12.cp /etc/rpc /etc/rpcOLD 13.sudo cp /etc/rpc /etc/rpcOLD 14.rm /etc/rpcOLD 15.sudo rm /etc/rpcOLD

Output	<pre> ubuntu@ubuntu:~\$ tail -n 5 /var/log/auth.log 2024-07-18T10:43:27.945081+00:00 ubuntu su[11327]: pam_unix(su:session): session opened for user root(uid=0) by ubuntu(uid=0) 2024-07-18T10:43:41.972474+00:00 ubuntu su[11327]: pam_unix(su:session): session closed for user root 2024-07-18T10:43:41.975250+00:00 ubuntu sudo: pam_unix(sudo:session): session closed for user root 2024-07-18T10:45:01.665624+00:00 ubuntu CRON[11343]: pam_unix(cron:session): session opened for user root(uid=0) by root(uid=0) 2024-07-18T10:45:01.673330+00:00 ubuntu CRON[11343]: pam_unix(cron:session): session closed for user root ubuntu@ubuntu:~\$ sudo tail -n 5 /var/log/auth.log 2024-07-18T10:43:41.975250+00:00 ubuntu sudo: pam_unix(sudo:session): session closed for user root 2024-07-18T10:45:01.665624+00:00 ubuntu CRON[11343]: pam_unix(cron:session): session opened for user root(uid=0) by root(uid=0) 2024-07-18T10:45:01.673330+00:00 ubuntu CRON[11343]: pam_unix(cron:session): session closed for user root 2024-07-18T10:45:38.040559+00:00 ubuntu sudo: ubuntu : TTY=pts/0 ; PWD=/home/ubuntu ; USER=root ; COMMAND=/usr/bin/tail -n 5 /var/log/auth.log 2024-07-18T10:45:38.050451+00:00 ubuntu sudo: pam_unix(sudo:session): session opened for user root(uid=0) by ubuntu(uid=1000) ubuntu@ubuntu:~\$ cp /etc/rpc/etc/rpcOLD cp: missing destination file operand after '/etc/rpc/etc/rpcOLD' Try 'cp --help' for more information. ubuntu@ubuntu:~\$ cp /etc/rpc/etc/rpcOLD </pre>
Que	<p>16. check the UID for root user, administrator and local users.</p> <p>17. Adduser user01.</p>
Commands	<p>10. id -u root getent group sudo id root</p> <p>11. sudo adduser user01</p>
Output	<pre> ubuntu@ubuntu:~\$ id -u root 0 ubuntu@ubuntu:~\$ getent group sudo sudo:x:27:ubuntu,installer ubuntu@ubuntu:~\$ id root uid=0(root) gid=0(root) groups=0(root) ubuntu@ubuntu:~\$ sudo adduser user01 info: Adding user `user01' ... info: Selecting UID/GID from range 1000 to 59999 ... info: Adding new group `user01' (1002) ... info: Adding new user `user01' (1002) with group `user01 (1002)' info: Creating home directory `/home/user01' ... info: Copying files from `/etc/skel' ... New password: BAD PASSWORD: The password is shorter than 8 characters Re-enter new password: </pre>

```

tematic
Retype new password:
passwd: password updated successfully
Changing the user information for user01
Enter the new value, or press ENTER for the default
    Full Name []: user01
    Room Number []: 1
    Work Phone []: 9988776655
    Home Phone []:
    Other []:
Is the information correct? [Y/n] y
info: Adding new user `user01' to supplemental / extra groups `user'
info: Adding user `user01' to group `users' ...
ubuntu@ubuntu:~$
```

Que.	18. Create the group group01 with the GID of 10000. 19. Create the group group02 20. Examine /etc/group to verify the supplemental group memberships.
Comma nd	18. sudo addgroup --gid 10000 group01 19. sudo addgroup group02 20. cat /etc/group
Output	<pre> ubuntu@ubuntu:~\$ sudo groupadd -g 1001 grp2 groupadd: GID '1001' already exists ubuntu@ubuntu:~\$ sudo groupadd -g 2001 grp2 ubuntu@ubuntu:~\$ sudo group add grp3 sudo: group: command not found ubuntu@ubuntu:~\$ sudo groupadd grp3 ubuntu@ubuntu:~\$ cat /etc/group root:x:0: daemon:x:1: bin:x:2: sys:x:3: adm:x:4:syslog,ubuntu,installer tty:x:5: disk:x:6: lp:x:7: mail:x:8: news:x:9: uucp:x:10: man:x:12: proxy:x:13: kmem:x:15: dialout:x:20:installer fax:x:21: voice:x:22:</pre>

Que.	
Command	
Output	<pre> fwupd-refresh:x:989: scanner:x:115:saned saned:x:116: geoclue:x:117: pipewire:x:118: polkitd:x:988: rtkit:x:119: colord:x:120: gdm:x:121: nm-openvpn:x:122: lxd:x:123:installer ubuntu:x:1000: sambashare:x:124:ubuntu gamemode:x:987: gnome-initial-setup:x:986: gnome-remote-desktop:x:985: installer:x:1001: user01:x:1002: group01sudo:x:10000: group02:x:10001: grp1:x:10002: grp2:x:2001: grp3:x:10003: ubuntu@ubuntu:~\$ </pre>

Que.	<p>21. Use the usermod -aG command to add a user to a supplementary group. Add user01 to the group created.</p> <p>22. Observe /etc/group and /etc/passwd</p>
Command	<p>21. sudo usermod -aG group01 user01</p> <p>22. cat /etc/group</p>

Output

```
ubuntu@ubuntu:~$ sudo usermod -aG grp31 user01
usermod: group 'grp31' does not exist
ubuntu@ubuntu:~$ sudo usermod -aG grp3 user01
ubuntu@ubuntu:~$ cat /etc/group
root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:syslog,ubuntu,installer
tty:x:5:
disk:x:6:
lp:x:7:
mail:x:8:
news:x:9:
uucp:x:10:
man:x:12:
proxy:x:13:
kmem:x:15:
dialout:x:20:installer
fax:x:21:
voice:x:22:

fwupd-refresh:x:989:
scanner:x:115:saned
saned:x:116:
geoclue:x:117:
pipewire:x:118:
polkitd:x:988:
rtkit:x:119:
colord:x:120:
gdm:x:121:
nm-openvpn:x:122:
lxd:x:123:installer
ubuntu:x:1000:
sambashare:x:124:ubuntu
gamemode:x:987:
gnome-initial-setup:x:986:
gnome-remote-desktop:x:985:
installer:x:1001:
user01:x:1002:
group01sudo:x:10000:
group02:x:10001:
grp1:x:10002:
grp2:x:2001:
grp3:x:10003:user01
ubuntu@ubuntu:~$
```

PART B

Control access to files

Que.	<ol style="list-style-type: none"> 1. Check the permission of files created. 2. Check the permission of directories created. 3. Set read and write permissions for others with numeric mode to file1.txt 4. Remove write permission for user, group and others to folder CE.
Command	<ol style="list-style-type: none"> 1. ls -l file2.txt 2. ls -ld CE 3. chmod 666 file1.txt 4. chmod a-w CE
Output	<pre>ubuntu@ubuntu:~\$ ls -l total 0 drwxrwxr-x 3 ubuntu ubuntu 60 Jun 27 18:29 CSPIT drwxr-xr-x 2 ubuntu ubuntu 60 Jun 27 15:43 Desktop drwxr-xr-x 2 ubuntu ubuntu 40 Jun 27 15:44 Documents drwxr-xr-x 2 ubuntu ubuntu 40 Jun 27 15:44 Downloads drwxr-xr-x 2 ubuntu ubuntu 40 Jun 27 15:44 Music drwxr-xr-x 2 ubuntu ubuntu 40 Jun 27 15:44 Pictures drwxr-xr-x 2 ubuntu ubuntu 40 Jun 27 15:44 Public drwxr-xr-x 2 ubuntu ubuntu 40 Jun 27 15:44 Templates drwxr-xr-x 2 ubuntu ubuntu 40 Jun 27 15:44 Videos drwx----- 5 ubuntu ubuntu 100 Jun 27 18:00 snap ubuntu@ubuntu:~\$ █ ubuntu@ubuntu:~/CSPIT\$ ls -l CE total 36 -rw-rw-r-- 1 ubuntu ubuntu 0 Jun 27 18:29 ce1.txt -rw-rw-r-- 1 ubuntu ubuntu 0 Jun 27 18:30 f3.txt -rw-rw-r-- 1 ubuntu ubuntu 43 Jun 27 18:31 file1.txt -rw-rw-r-- 1 ubuntu ubuntu 38 Jun 27 18:50 file1_no_digits.txt -rw-rw-r-- 1 ubuntu ubuntu 55 Jun 27 18:32 file2.txt -rw-rw-r-- 1 ubuntu ubuntu 56 Jun 27 18:37 file4.txt -rw-rw-r-- 1 ubuntu ubuntu 48 Jun 27 18:39 file5.txt -rw-rw-r-- 1 ubuntu ubuntu 54 Jun 27 18:47 file6.txt -rw-rw-r-- 1 ubuntu ubuntu 60 Jun 27 18:47 file7.txt -rw-rw-r-- 1 ubuntu ubuntu 118 Jun 27 18:52 file8.txt -rw-rw-r-- 1 ubuntu ubuntu 215 Jul 11 15:57 file9.txt ubuntu@ubuntu:~/CSPIT\$ cd CE ubuntu@ubuntu:~/CSPIT/CE\$ chmod 66 file1.txt] chmod: cannot access 'file1.txt]': No such file or directory ubuntu@ubuntu:~/CSPIT/CE\$ chmod 66 file1.txt ubuntu@ubuntu:~/CSPIT/CE\$ cd ~ ubuntu@ubuntu:~\$ chmod a-w CE chmod: cannot access 'CE': No such file or directory ubuntu@ubuntu:~\$ cd CSPIT ubuntu@ubuntu:~/CSPIT\$ chmod a-w CE ubuntu@ubuntu:~/CSPIT\$ █</pre>

Que.	<ol style="list-style-type: none"> 5. Create a directory 5CE under CE. Observe the response. 6. Set read, write and execute permissions for user, group and others to 5CE. 7. Set read and execute permission for group and no permission for other to file2.txt. 8. Change the ownership of file to user01 9. Change the group ownership of file to group01 10. Change the ownership of both group and user at the same time.
-------------	--

Command	5. mkdir CE/5CE 6. chmod 777 CE/5CE 7. chmod 750 file2.txt 8. sudo chown user01 file2.txt 9. sudo chown :group01 file2.txt 10. sudo chown user01:group01 file_name
Output	<pre>ubuntu@ubuntu:~/CSPIT\$ sudo mkdir 5CE ubuntu@ubuntu:~/CSPIT\$ cd CE 5 ubuntu@ubuntu:~/CSPIT/CE\$ chmod 750 file2.txt ubuntu@ubuntu:~/CSPIT/CE\$ sudo chown user01 file2.txt ubuntu@ubuntu:~/CSPIT/CE\$ ls -l file2.txt -rwxr-x--- 1 user01 ubuntu 55 Jun 27 18:32 file2.txt ubuntu@ubuntu:~/CSPIT/CE\$ sudo chown :grp1 file2.txt ubuntu@ubuntu:~/CSPIT/CE\$ ls -l file2.txt -rwxr-x--- 1 user01 grp1 55 Jun 27 18:32 file2.txt ubuntu@ubuntu:~/CSPIT/CE\$ sudo chown user01:grp1 file2.txt ubuntu@ubuntu:~/CSPIT/CE\$ la -l file2.txt -rwxr-x--- 1 user01 grp1 55 Jun 27 18:32 file2.txt ubuntu@ubuntu:~/CSPIT/CE\$</pre>
Que.	11. Set the special permissions on directory. a. The <i>setuid</i> permission on an executable file means that commands run as the user owning the file, not as the user that ran the command. One example is the passwd command:run ls -l /usr/bin/passwd b. The special permission <i>setgid</i> on a directory means that files created in the directory inherit their group ownership from the directory, rather than inheriting it from the creating user. run ls -ld /run/log/journal c. the <i>sticky bit</i> for a directory sets a special restriction on deletion of files. Only the owner of the file (and root) can delete files within the directory. run ls -ld /tmp 12. Set the setuid, setgid and sticky bit for different files and perform the operations accordingly.
Comma nd	11. a) ls -l /usr/bin/passwd b) ls -ld /run/log/journal c) ls -ld /tmp 12. sudo chmod u+s file_name sudo chmod g+s directory_name sudo chmod +t directory_name

Output

```
ubuntu@ubuntu:~/CSPIT/CE$ ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 64152 Apr  9 12:31 /usr/bin/passwd
ubuntu@ubuntu:~/CSPIT/CE$ ls -ld /run/log/journal
drwxr-sr-x+ 2 root systemd-journal 40 Jun 27 15:43 /run/log/journal
ubuntu@ubuntu:~/CSPIT/CE$ ls -ld /tmp
drwxrwxrwt 23 root root 500 Aug  8 12:48 /tmp
ubuntu@ubuntu:~/CSPIT/CE$ sudo chmod u+s file2.txt
ubuntu@ubuntu:~/CSPIT/CE$ ls -l file2.txt
-rwsr-x--- 1 user01 grp1 55 Jun 27 18:32 file2.txt
ubuntu@ubuntu:~/CSPIT/CE$ sudo chmod g+s student
chmod: cannot access 'student': No such file or directory
ubuntu@ubuntu:~/CSPIT/CE$ sudo chmod g+s CE
chmod: cannot access 'CE': No such file or directory
ubuntu@ubuntu:~/CSPIT/CE$ sudo chmod g+s 5CE
ubuntu@ubuntu:~/CSPIT/CE$ ls -ld 5CE
drwxr-sr-x 2 root root 40 Aug  8 13:08 5CE
ubuntu@ubuntu:~/CSPIT/CE$ sudo chmod +t 5CE
ubuntu@ubuntu:~/CSPIT/CE$ ls -ld 5CE
drwxr-sr-t 2 root root 40 Aug  8 13:08 5CE
ubuntu@ubuntu:~/CSPIT/CE$
```

- | | |
|---------------------|---|
| Que. | <ol style="list-style-type: none">13. Display the current value of shell's mask.14. Check the permission of directories.15. Check the permission of files.16. Set the umask to 542.17. Check the permission of files and directories.18. Try to open the file and directory created.19. Try to open the file as other user. |
| Comma
nd | <ol style="list-style-type: none">13. umask14. ls -ld directory_name15. ls -l file_name16. umask 54217. touch new_file
mkdir new_directory
ls -l new_file
ls -ld new_directory18. cat new_file
cd new_directory19. su another_user
cat new_file |

Output

```
ubuntu@ubuntu:~/CSPIT/CE$ sudo mkdir newdir
ubuntu@ubuntu:~/CSPIT/CE$ ls -l ce1.txt
-rw-rw-r-- 1 ubuntu ubuntu 0 Aug  8 13:20 ce1.txt
ubuntu@ubuntu:~/CSPIT/CE$ ls -ld mkdir
ls: cannot access 'mkdir': No such file or directory
ubuntu@ubuntu:~/CSPIT/CE$ ls -ld newdir

drwxr-xr-x 2 root root 40 Aug  8 13:20 newdir
ubuntu@ubuntu:~/CSPIT/CE$ cat ce1.txt
ubuntu@ubuntu:~/CSPIT/CE$ cd newdir
ubuntu@ubuntu:~/CSPIT/CE/newdir$ su another_user
su: user another_user does not exist or the user entry does not contain the required fields
ubuntu@ubuntu:~/CSPIT/CE/newdir$ adduser another_user
fatal: Only root may add a user or group to the system.
ubuntu@ubuntu:~/CSPIT/CE/newdir$ su user01
Password:
user01@ubuntu:/home/ubuntu/CSPIT/CE/newdir$ cat ce1.txt
```

```
ubuntu@ubuntu:~/CSPIT/CE$ cd newdir
ubuntu@ubuntu:~/CSPIT/CE/newdir$ su user01
Password:
user01@ubuntu:/home/ubuntu/CSPIT/CE/newdir$ cat ce1.txt
user01@ubuntu:/home/ubuntu/CSPIT/CE/newdir$ cat ce1.txt
hello students . hello students
user01@ubuntu:/home/ubuntu/CSPIT/CE/newdir$
```

Practical 3

Aim: Process Management

1. Managing and monitoring linux processes
2. Control Services and Daemons
3. Improve Command Line productivity

Commands for reference:

Process: top, ps, kill, pkill, w, lscpu

Control Services and Daemons: systemctl with parameters start, stop, restart, enabledisable, is-active, is-enabled and is-failed service

I/O Redirection (<, >, >>), Pipe (|)

Part-A

Monitoring and Managing Linux Processes

Que.	1. List down all processes with their states sorted by their CPU Usage. Identify current running process.
Command	ps
Output	<pre>ubuntu@ubuntu:~\$ ps PID TTY TIME CMD 10460 pts/9 00:00:00 bash 10474 pts/9 00:00:00 top 10477 pts/9 00:00:00 ps</pre>

Que.	2. List down all processes associated with current user.
Command	ps -u
Output	<pre>ubuntu@ubuntu: \$ ps -u USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND ubuntu 1691 0.0 0.2 244800 5504 tty2 Ssl+ 01:55 0:00 /usr/libexec/gdm-x-session --run-script env GNOME_SHE ubuntu 1703 5.3 2.9 347000 59656 tty2 RL+ 01:55 12:38 /usr/lib/xorg/Xorg vt2 -displayfd 3 -auth /run/user/1 ubuntu 1935 0.0 0.4 307288 8960 tty2 Sl+ 01:55 0:00 /usr/libexec/gnome-session-binary --session=ubuntu ubuntu 10460 0.0 0.2 20100 5120 pts/9 Ss 05:44 0:00 bash ubuntu 10474 0.0 0.2 23640 5888 pts/9 T 05:45 0:00 top ubuntu 10489 200 0.2 22720 4480 pts/9 R+ 05:53 0:00 ps -u</pre>

Que.	3. List down all processes associated with their terminal and their states. Identify current running process.
Command	Ps -t
Output	<pre>ubuntu@ubuntu:~\$ ps -t PID TTY STAT TIME COMMAND 10460 pts/9 Ss 0:00 bash 10474 pts/9 T 0:00 top 10488 pts/9 R+ 0:00 ps -t</pre>

Que.	4. Compare the output of “ps lx” and “ps l” commands
Command	Ps lx Ps l
Output	<pre>ubuntu@ubuntu:~\$ ps lx F UID PID PPID PRI NI VSZ RSS WCHAN STAT TTY TIME COMMAND 4 1000 1649 1 20 0 21292 8704 - Ss ? 0:03 /usr/lib/systemd/systemd --user 5 1000 1652 1649 20 0 21456 3468 - S ? 0:00 (sd-pam) 4 1000 1670 1649 20 0 124104 11072 ep_pol Ssl ? 0:13 /usr/bin/pipewire 4 1000 1671 1649 20 0 106804 3840 ep_pol Ssl ? 0:00 /usr/bin/pipewire -c filter-chain.conf 4 1000 1672 1649 20 0 416368 12544 do_pol Ssl ? 0:04 /usr/bin/wireplumber 4 1000 1674 1649 20 0 130632 11992 ep_pol Ssl ? 0:06 /usr/bin/pipewire-pulse 4 1000 1675 1649 20 0 325596 7168 do_pol Slsl ? 0:00 /usr/bin/gnome-keyring-daemon --foreground --config-file=/usr/share/gnome-session/gnome-keyring-daemon.conf 4 1000 1680 1649 20 0 10916 5260 ep_pol Ss ? 0:05 /usr/bin/dbus-daemon --session --address=system bus 4 1000 1691 1641 20 0 244800 5504 do_pol Ssl+ tty2 0:00 /usr/libexec/gdm-x-session --run-script env GDMSESSION=gnome 4 1000 1703 1691 20 0 347000 59656 - RL+ tty2 12:39 /usr/lib/xorg/Xorg vt2 -displayfd 3 -auth /run/user/1000/.Xauthority 4 1000 1935 1691 20 0 307288 8960 do_pol Sls+ tty2 0:00 /usr/libexec/gnome-session-binary --session=ubuntu 4 1000 2128 1649 20 0 382984 7040 do_pol Ssl ? 0:00 /usr/libexec/at-spi-bus-launcher 4 1000 2128 2128 20 0 9608 3840 ep_pol S ? 0:00 /usr/bin/dbus-daemon --config-file=/usr/share/dbus-1/system.conf 4 1000 2162 1649 20 0 162644 5504 do_pol Ssl ? 0:00 /usr/libexec/gcr-ssh-agent --base-dir /run/user/1000/.gcr 4 1000 2163 1649 20 0 100636 4736 do_pol Ssl ? 0:00 /usr/libexec/gnome-session-ctl --monitor 4 1000 2180 1649 20 0 323360 6016 do_pol Ssl ? 0:00 /usr/libexec/gvfsd 4 1000 2194 1649 20 0 468812 5888 futex_ Slsl ? 0:00 /usr/libexec/gvfsd-fuse /run/user/1000/gvfs -f fuse 4 1000 2216 1649 20 0 742588 10624 do_pol Ssl ? 0:00 /usr/libexec/gnome-session-binary --session=ubuntu 4 1000 2244 1649 20 0 4867936 347616 do_pol Ssl ? 0:28 /usr/bin/gnome-shell</pre> <pre>ubuntu@ubuntu:~\$ ps l F UID PID PPID PRI NI VSZ RSS WCHAN STAT TTY TIME COMMAND 4 1000 1691 1641 20 0 244800 5504 do_pol Ssl+ tty2 0:00 /usr/libexec/gdm-x-session --run-script env GDMSESSION=gnome 4 1000 1703 1691 20 0 347000 59656 - RL+ tty2 12:37 /usr/lib/xorg/Xorg vt2 -displayfd 3 -auth /run/user/1000/.Xauthority 4 1000 1935 1691 20 0 307288 8960 do_pol Sls+ tty2 0:00 /usr/libexec/gnome-session-binary --session=ubuntu 4 1000 10460 10447 20 0 20100 5120 do_wai Ss pts/9 0:00 bash 4 1000 10474 10460 20 0 23640 5888 do_sig T pts/9 0:00 top 0 1000 10487 10460 20 0 22752 4352 - R+ pts/9 0:00 ps l</pre>

Que.	5. List down all the names and numbers of all available signals
Command	kill -l
Output	<pre>ubuntu@ubuntu:~\$ kill -l 1) SIGHUP 2) SIGINT 3) SIGQUIT 4) SIGILL 5) SIGTRAP 6) SIGABRT 7) SIGBUS 8) SIGFPE 9) SIGKILL 10) SIGUSR1 11) SIGSEGV 12) SIGUSR2 13) SIGPIPE 14) SIGALRM 15) SIGTERM 16) SIGSTKFLT 17) SIGCHLD 18) SIGCONT 19) SIGSTOP 20) SIGTSTP 21) SIGTTIN 22) SIGTTOU 23) SIGURG 24) SIGXCPU 25) SIGXFSZ 26) SIGVTALRM 27) SIGPROF 28) SIGWINCH 29) SIGIO 30) SIGPWR 31) SIGSYS 34) SIGRTMIN 35) SIGRTMIN+1 36) SIGRTMIN+2 37) SIGRTMIN+3 38) SIGRTMIN+4 39) SIGRTMIN+5 40) SIGRTMIN+6 41) SIGRTMIN+7 42) SIGRTMIN+8 43) SIGRTMIN+9 44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13 48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12 53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9 56) SIGRTMAX-8 57) SIGRTMAX-7 58) SIGRTMAX-6 59) SIGRTMAX-5 60) SIGRTMAX-4 61) SIGRTMAX-3 62) SIGRTMAX-2 63) SIGRTMAX-1 64) SIGRTMAX</pre>

Que.	6. Run the “sleep 10000” in background.
Command	sleep 10000 &
Output	<pre>ubuntu@ubuntu:~\$ sleep 10000 & [2] 10594</pre>

Que.	7.Check the PID of sleep process and kill it using PID.
Command	pgrep sleep kill (PID)
Output	<pre>ubuntu@ubuntu:~\$ pgrep sleep 10594 ubuntu@ubuntu:~\$ kill 10594</pre>

Que.	8.Apply w command and observe the output
Command	w
Output	<pre>ubuntu@ubuntu:~\$ w 05:23:12 up 4:07, 1 user, load average: 0.05, 0.09, 0.12 USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT ubuntu tty2 - 22Jul24 13days 12:45 0.21s /usr/libexec/gnome-session-binary --session=ubuntu [2]- Terminated sleep 10000</pre>

Que.	9. Open the firefox browser. Check the processes associated with firefox
Command	top
Output	<pre>ubuntu@ubuntu:~\$ top top - 05:26:26 up 4:10, 1 user, load average: 1.10, 0.61, 0.31 Tasks: 253 total, 2 running, 243 sleeping, 8 stopped, 0 zombie %Cpu(s): 7.8 us, 8.6 sy, 0.0 ni, 83.3 id, 0.0 wa, 0.0 hi, 0.3 si, 0.0 st MiB Mem : 1967.2 total, 82.4 free, 1774.0 used, 508.9 buff/cache MiB Swap: 0.0 total, 0.0 free, 0.0 used. 193.2 avail Mem PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND 2244 ubuntu 20 0 4871596 315980 39304 R 75.5 15.7 11:52.76 gnome-shell 1703 ubuntu 20 0 378732 88196 37004 S 4.6 4.4 12:57.11 Xorg 10599 ubuntu 20 0 2938092 265500 129120 S 0.7 13.2 0:07.71 firefox 11422 ubuntu 20 0 23616 5632 3456 R 0.7 0.3 0:00.07 top</pre>

Que.	10. Kill all processes associated with firefox by its name.
Command	Pkill firefox
Output	<pre>ubuntu@ubuntu:~\$ pkill firefox</pre>

Que.	11. Give the difference between kill and pkill.
Output	kill: Sends a signal to a process specified by its PID. pkill: Sends a signal to all processes that match a specified name or other attributes.

Que.	12. Run “lscpu” command and observe the output.
Command	lscpu
Output	<pre>ubuntu@ubuntu:~\$ lscpu Architecture: x86_64 CPU op-mode(s): 32-bit, 64-bit Address sizes: 48 bits physical, 48 bits virtual Byte Order: Little Endian CPU(s): 4 On-line CPU(s) list: 0-3 Vendor ID: AuthenticAMD Model name: AMD Ryzen 7 6800H with Radeon Graphics CPU family: 25 Model: 68 Thread(s) per core: 1 Core(s) per socket: 4 Socket(s): 1 Stepping: 1 BogoMIPS: 6387.99 Flags: fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge m ca cmov pat pse36 clflush mmx fxsr sse sse2 ht syscall nx mmxext fxsr_opt rdtscp lm constant_tsc rep_good no pl nonstop_tsc cpuid extd_apicid tsc_known_freq pnpi pc lmulqdq ssse3 cx16 sse4_1 sse4_2 movbe popcnt aes rdrai nd hypervisor lahf_lm cmp_legacy cr8_legacy abm sse4a misalignsse 3dnowprefetch vmmcall fsgsbase bmi1 bmi2 i nvpcid rdseed clflushopt arat Virtualization features: Hypervisor vendor: KVM Virtualization type: full Caches (sum of all): L1d: 128 KiB (4 instances) L1i: 128 KiB (4 instances) L2: 2 MiB (4 instances) L3: 64 MiB (4 instances)</pre>

PART B

Control Services and daemons

Que.	1. List all services on your system.(<code>systemctl list-units --type=service</code>)
Command	<code>systemctl list-units --type=service</code>
Output	<pre>ubuntu@ubuntu:~\$ systemctl list-units --type=service UNIT LOAD ACTIVE SUB accounts-daemon.service loaded active ru> alsa-restore.service loaded active ex> apport.service loaded active ex> avahi-daemon.service loaded active ru> blk-availability.service loaded active ex> casper-md5check.service loaded active ex> cloud-config.service loaded active ex> cloud-final.service loaded active ex> cloud-init-local.service loaded active ex> cloud-init.service loaded active ex> colord.service loaded active ru> console-setup.service loaded active ex> cron.service loaded active ru> cups-browsed.service loaded active ru> cups.service loaded active ru> dbus.service loaded active ru> finalrd.service loaded active ex> fwupd.service loaded active ru> gdm.service loaded active ru> lines 1-20... skipping...</pre>

Que.	2. Check whether the ssh service is active or not. (<code>sudo systemctl status service_name</code>)
Command	<code>sudo systemctl status kerneloops.service</code>
Output	<pre>ubuntu@ubuntu: \$ sudo systemctl status kerneloops.service ● kerneloops.service - Tool to automatically collect and submit kernel crash signatures Loaded: loaded (/usr/lib/systemd/system/kerneloops.service; enabled; preset: enabled) Active: active (running) since Mon 2024-08-12 09:32:41 UTC; 58min ago Tasks: 2 (limit: 2266) Memory: 1.4M (peak: 1.8M) CPU: 107ms CGroup: /system.slice/kerneloops.service └─1616 /usr/sbin/kerneloops --test ├─1619 /usr/sbin/kerneloops Aug 12 09:32:41 ubuntu systemd[1]: Starting kerneloops.service - Tool to automatically collect and submit kernel crash > Aug 12 09:32:41 ubuntu systemd[1]: kerneloops.service: Found left-over process 1616 (kerneloops) in control group while > Aug 12 09:32:41 ubuntu systemd[1]: kerneloops.service: This usually indicates unclean termination of a previous run, or > Aug 12 09:32:41 ubuntu (rneloops)[1617]: kerneloops.service: Referenced but unset environment variable evaluates to an > Aug 12 09:32:41 ubuntu systemd[1]: Started kerneloops.service - Tool to automatically collect and submit kernel crash s> lines 1-15/15 (END)</pre>

Que.	3. If the package is not available, install ssh package (sudo apt-get install ssh)
Command	sudo apt-get install ssh
Output	<pre>ubuntu@ubuntu:~\$ sudo apt-get install ssh Reading package lists... Done Building dependency tree... Done Reading state information... Done The following additional packages will be installed: ncurses-term openssh-server openssh-sftp-server ssh-import-id Suggested packages: molly-guard monkeysphere ssh-askpass The following NEW packages will be installed: ncurses-term openssh-server openssh-sftp-server ssh ssh-import-id 0 upgraded, 5 newly installed, 0 to remove and 0 not upgraded. Need to get 837 kB of archives. After this operation, 6809 kB of additional disk space will be used. Do you want to continue? [Y/n] y Get:1 http://archive.ubuntu.com/ubuntu noble/main amd64 openssh-sftp-server amd64 1:9.6p1-3ubuntu13 [37.3 kB] Get:2 http://archive.ubuntu.com/ubuntu noble/main amd64 openssh-server amd64 1:9.6p1-3ubuntu13 [510 kB] Get:3 http://archive.ubuntu.com/ubuntu noble/main amd64 ncurses-term all 6.4+20240113-1ubuntu2 [275 kB] Get:4 http://archive.ubuntu.com/ubuntu noble/main amd64 ssh all 1:9.6p1-3ubuntu13 [4650 B] Get:5 http://archive.ubuntu.com/ubuntu noble/main amd64 ssh-import-id all 5.11-0ubuntu2 [10.0 kB] Fetched 837 kB in 3s (250 kB/s) Preconfiguring packages ... Selecting previously unselected package openssh-sftp-server. (Reading database ... 210703 files and directories currently installed.) Preparing to unpack .../openssh-sftp-server_1%3a9.6p1-3ubuntu13_amd64.deb ... Unpacking openssh-sftp-server (1:9.6p1-3ubuntu13) ... Selecting previously unselected package openssh-server. Preparing to unpack .../openssh-server_1%3a9.6p1-3ubuntu13_amd64.deb ... Unpacking openssh-server (1:9.6p1-3ubuntu13) ... Selecting previously unselected package ncurses-term. Preparing to unpack .../ncurses-term_6.4+20240113-1ubuntu2_all.deb ... Unpacking ncurses-term (6.4+20240113-1ubuntu2) ... Selecting previously unselected package ssh.</pre>

Que.	4. If the service is available and active, check the process state usng ps -p PID
Command	ps -p 3528
Output	<pre>ubuntu@ubuntu:~\$ ps -p 3528 PID TTY TIME CMD 3528 ? 00:00:01 gnome-calendar</pre>

Que.	5. Add the firewall rule to allow remote service using ssh(sudo ufw allow ssh)
Command	sudo ufw allow ssh
Output	<pre>ubuntu@ubuntu:~\$ sudo ufw allow ssh Skipping adding existing rule Skipping adding existing rule (v6)</pre>

Que.	6. Check your IP address
Command	ifconfig
Output	<pre>ubuntu@ubuntu:~\$ ifconfig enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500 inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255 inet6 fe80::a00:27ff:fe59:af prefixlen 64 scopeid 0x20<link> ether 08:00:27:fe:59:af txqueuelen 1000 (Ethernet) RX packets 2183 bytes 2835785 (2.8 MB) RX errors 0 dropped 0 overruns 0 frame 0 TX packets 880 bytes 111595 (111.5 KB) TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0 lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536 inet 127.0.0.1 netmask 255.0.0.0 inet6 ::1 prefixlen 128 scopeid 0x10<host> loop txqueuelen 1000 (Local Loopback) RX packets 2651 bytes 240481 (240.4 KB) RX errors 0 dropped 0 overruns 0 frame 0 TX packets 2651 bytes 240481 (240.4 KB) TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0</pre>

Que.	7. Access another user terminal using ssh
Command	ssh shiv@127.0.1.1
Output	<pre>ubuntu@ubuntu:~\$ sudo service ssh restart ubuntu@ubuntu:~\$ ssh shiv@127.0.1.1 The authenticity of host '127.0.1.1 (127.0.1.1)' can't be established. ED25519 key fingerprint is SHA256:HIR69+XZc0J3UC/dU3gRki21tSNwlx9fIBB+dj80Zoo. This key is not known by any other names. Are you sure you want to continue connecting (yes/no/[fingerprint])? yes Warning: Permanently added '127.0.1.1' (ED25519) to the list of known hosts. shiv@127.0.1.1's password: Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-31-generic x86_64) * Documentation: https://help.ubuntu.com * Management: https://landscape.canonical.com * Support: https://ubuntu.com/pro The programs included with the Ubuntu system are free software; the exact distribution terms for each program are described in the individual files in /usr/share/doc/*copyright. Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law. shiv@ubuntu:~\$ s</pre>

Que.	8. Stop the service and check the status
Command	sudo systemctl stop ssh
Output	<pre>ubuntu@ubuntu:~\$ sudo systemctl stop ssh Stopping 'ssh.service', but its triggering units are still active: ssh.socket</pre>

Que.	9. Disable the service and check the status
Command	sudo systemctl disable ssh
Output	<pre>ubuntu@ubuntu:~\$ sudo systemctl disable ssh Synchronizing state of ssh.service with SysV service script with /usr/lib/systemd/systemd-sysv-install. Executing: /usr/lib/systemd/systemd-sysv-install disable ssh Disabling 'ssh.service', but its triggering units are still active: ssh.socket</pre>

Que.	10. Enable it again and check the status
Command	sudo systemctl enable ssh
Output	<pre>ubuntu@ubuntu:~\$ sudo systemctl enable ssh Synchronizing state of ssh.service with SysV service script with /usr/lib/systemd/systemd-sysv-install. Executing: /usr/lib/systemd/systemd-sysv-install enable ssh Created symlink /etc/systemd/system/sshd.service → /usr/lib/systemd/system/ssh.service. Created symlink /etc/systemd/system/multi-user.target.wants/ssh.service → /usr/lib/systemd/system/ssh.service.</pre>

Que.	11. Restart the service and check the status
Command	sudo systemctl restart ssh
Output	ubuntu@ubuntu:~\$ sudo systemctl restart ssh

Que.	12. Observe the analyze the output of below mentioned command 1. systemctl is-active ssh 2. systemctl is-enabled ssh 3. systemctl is-failed ssh
Command	1. systemctl is-active ssh 2. systemctl is-enabled ssh 3. systemctl is-failed ssh
Output	<pre>ubuntu@ubuntu:~\$ systemctl is-active ssh active ubuntu@ubuntu:~\$ systemctl is-enabled ssh enabled ubuntu@ubuntu:~\$ systemctl is-failed ssh active</pre>

PART C

Improve Command Line Productivity

I/o Redirection

Que.	1. Create a file named “newfile.txt” and insert a text into created file as follow: “The operating system is a system program that serves as an interface between the computing system and the end-user.”
Command	touch newfile.txt Cat > newfile.txt
Output	<pre>ubuntu@ubuntu:~/DEPSTAR/CE\$ touch newfile.txt ubuntu@ubuntu:~/DEPSTAR/CE\$ cat > newfile.txt The Operating system is system program that serves as an interface between the computing system and end-user</pre>

Que.	2. Redirect the output of “newfile.txt” file to file “new.txt” using command. 3. Type command cat, then enter key and enter some text. Observe the output
Command	Cat newfile.txt > new.txt cat
Output	<pre>ubuntu@ubuntu:~/DEPSTAR/CE\$ cat newfile.txt > new.txt ubuntu@ubuntu:~/DEPSTAR/CE\$ cat Examples of Operating system: Windows, iOS Examples of Operating system: Windows, iOS ^Z [1]+ Stopped cat</pre>

Que.	4. Type command i) cat <newfile.txt ii) cat newfile.txt. Interpret the output in both cases.
Command	i. cat < newfile.txt ii. cat newfile.txt
Output	<pre>ubuntu@ubuntu:~/DEPSTAR/CE\$ cat < newfile.txt The Operating system is system program that serves as an interface between the computing system and end-user ubuntu@ubuntu:~/DEPSTAR/CE\$ cat newfile.txt The Operating system is system program that serves as an interface between the computing system and end-user</pre>

Que.	5. Type command cat – and enter any text.
Command	cat -
Output	<pre>ubuntu@ubuntu:~/DEPSTAR/CE\$ cat - Hello Hello ^Z [3]+ Stopped cat -</pre>

Que.	6. Type command i) cat < and > at once to redirect the output of one file to another.
Command	cat < newfile.txt > new.txt
Output	<pre>ubuntu@ubuntu:~/DEPSTAR/CE\$ cat < newfile.txt > new.txt</pre>

Que.	7. Summarize the use of cat command with redirection operator based on your done exercise.
Output	<p>cat file: Displays the contents of file.</p> <p>cat < file: Reads from file and outputs the contents.</p> <p>cat > file: Takes input from the keyboard and writes it to file. Press Ctrl+D to end the input.</p> <p>cat file1 > file2: Redirects the contents of file1 to file2, overwriting file2.</p> <p>cat file1 >> file2: Appends the contents of file1 to file2.</p> <p>cat -: Reads from standard input and echoes the input back.</p>

Que.	8. Try following command and interpret the output: a. ls >filelist b. cat newfile.txt new.txt >> report c. cat newfile.txt > newfile.txt d. date; who e. date; who>logfile f. (date; who) > logfile
Command	a. ls >filelist b. cat newfile.txt new.txt >> report c. cat newfile.txt > newfile.txt d. date; who e. date; who>logfile f. (date; who) > logfile
Output	<pre>ubuntu@ubuntu:~/DEPSTAR/CE\$ ls > filelist ubuntu@ubuntu:~/DEPSTAR/CE\$ cat newfile.txt new.txt >> report ubuntu@ubuntu:~/DEPSTAR/CE\$ cat newfile.txt > newfile.txt ubuntu@ubuntu:~/DEPSTAR/CE\$ date; who Sun Aug 4 05:51:07 UTC 2024 ubuntu seat0 2024-07-22 03:57 (login screen) ubuntu :0 2024-07-22 03:57 (:0) ubuntu pts/1 2024-07-22 04:47 ubuntu pts/2 2024-07-22 04:49 ubuntu pts/4 2024-07-28 05:33 ubuntu pts/6 2024-07-28 05:34 ubuntu pts/8 2024-07-28 05:43 ubuntu@ubuntu:~/DEPSTAR/CE\$ date; who > logfile Sun Aug 4 05:51:22 UTC 2024 ubuntu@ubuntu:~/DEPSTAR/CE\$ (date;who) > logfile</pre>

Piping

Que.	1. ls wc -l
Command	ls wc -l
Output	<code>ubuntu@ubuntu:~/DEPSTAR/CE\$ ls wc -l 17</code>

Que.	2.ls less
Command	ls less
Output	<pre>22DCE082_20240727.txt ce.txt f3.txt file1.txt file2.txt file4.txt file5.txt file6.txt file7.txt file8.txt file9.txt filelist logfile new.txt newfile.txt no_digit.txt report (END)</pre>

Que.	3. store the value of count in file named “countfile” using pipeline. 4. Try command who sort and observe the output.
Command	Ls wc -l > countfile who sort
Output	<code>ubuntu@ubuntu:~/DEPSTAR/CE\$ ls wc -l > countfile ubuntu@ubuntu:~/DEPSTAR/CE\$ who sort ubuntu :0 2024-07-22 03:57 (:0) ubuntu pts/1 2024-07-22 04:47 ubuntu pts/2 2024-07-22 04:49 ubuntu pts/4 2024-07-28 05:33 ubuntu pts/6 2024-07-28 05:34 ubuntu pts/8 2024-07-28 05:43 ubuntu seat0 2024-07-22 03:57 (login screen)</code>

Que.	5. Store the sorted output in file named “sortedlist”
Command	Sort newfile.txt > new.txt
Output	<code>ubuntu@ubuntu:~/DEPSTAR/CE\$ sort newfile.txt > new.txt</code>

Que.	6. Try cal 1996 head -10
Command	cal 1996 head-10
Output	<pre>ubuntu@ubuntu:~\$ cal 1996 head-10 Command 'cal' not found, but can be installed with: sudo apt install ncal head-10: command not found</pre>

Que.	7. who sort – logfile > newfile
Command	who sort – logfile > newfile
Output	<pre>ubuntu@ubuntu:~/DEPSTAR/CE\$ who sort - logfile > newfile</pre>

CONCLUSION:

Through this practical, I learned about managing Linux processes and controlling services. I also learned about how to monitor and manage processes, control system services, and use commands like I/O redirection and piping to work more effectively in the terminal.

PROBLEMS FACED:

Practical 4

Aim: Study of Linux File System

Que.	1. The File Hierarchy Standard (FHS) is a specification that defines the file system hierarchy of a Linux OS. Illustrate about the use of all directories under the “\” as given in the figure.
Output	<p>/boot Contains all the files required for booting the system, such as the Linux kernel, initial RAM disk (initrd), and bootloader configuration files (e.g., GRUB). It is essential for system startup.</p> <p>/bin (Binaries) Holds essential command binaries (executables) like basic commands needed by all users such as ls, cp, mv, cat, etc. These binaries are crucial during system boot or single-user mode when other parts of the filesystem (like /usr) may not be mounted yet.</p> <p>/dev (Devices) Contains special device files that represent hardware devices (such as hard drives, USBs, keyboards, etc.) and pseudo-devices (e.g., /dev/null). Devices can be interacted with as if they were files.</p> <p>/etc (Configuration) This directory contains system-wide configuration files, scripts, and settings. For example, /etc/passwd contains user account information, and /etc/fstab defines filesystem mounts.</p> <p>/lib (Libraries) Houses shared libraries needed by the binaries in /bin and /sbin. These libraries (similar to DLL files in Windows) are necessary for the system's basic functioning.</p> <p>/proc (Processes)</p>

A virtual filesystem that provides information about system processes and other kernel-related information. Files in /proc don't actually exist on disk; they are generated by the system and represent live system information. For example, /proc/cpuinfo contains CPU details.

/root

This is the home directory of the root user (the superuser). Unlike normal users whose home directories are in /home, the root user's directory is separate, under /.

/sbin (System Binaries)

Contains essential system binaries that are typically used by the root user for system administration tasks (e.g., fsck, shutdown, mount). These commands are important for the maintenance and repair of the system.

/tmp (Temporary)

Used for storing temporary files created by programs. The files in /tmp are typically cleared out either at boot time or periodically by the system.

/usr (User Programs)

Holds user programs and utilities. It contains many important subdirectories:

- a. /usr/bin: Non-essential user commands.
- b. /usr/sbin: Non-essential system binaries for administrative tasks.
- c. /usr/lib: Libraries for binaries in /usr/bin and /usr/sbin.
- d. /usr/share: Architecture-independent files such as icons, documentation, and shared libraries.
- e. /usr/local: Software and scripts that are not part of the distribution but are installed manually by the system administrator.

/var (Variable Files)

Stores files that are expected to change frequently. Common subdirectories include:

- a. /var/log: System log files.
- b. /var/spool: Directories for queued work, like print jobs and email.
- c. /var/lib: Holds state information that programs need to preserve between reboots.

Que.	2. List out files in your directory.
Command	ls
Output	<pre>ubuntu@ubuntu:~/Documents\$ ls ABC S1.txt a1.txt s1.sh s2.sh s4.sh s6.sh s7.sh.save s9.sh S1.sh S2.sh f1.txt s10.sh s3.sh s5.sh s7.sh s8.sh</pre>

Que.	3. Create a hard link to one of the file exist in your directory. (ln [original file] [link name])
Command	ln a1.txt file.txt
Output	<pre>ubuntu@ubuntu:~/Documents\$ ln a1.txt file.txt</pre>

Que.	4. Apply ls -l and check whether the link is created or not. Also check the size of linked file created.
Command	ls -l
Output	<pre>ubuntu@ubuntu:~/Documents\$ ls -l total 68 drwxrwxr-x 2 ubuntu ubuntu 60 Aug 12 11:05 ABC -rwxrwxrwx 1 ubuntu ubuntu 102 Sep 2 10:48 S1.sh -rwxrwxr-x 1 ubuntu ubuntu 14 Oct 8 07:08 S1.txt -rwxrwxr-x 1 ubuntu ubuntu 29 Sep 2 09:59 S2.sh -rwxrwxr-x 2 ubuntu ubuntu 20 Oct 8 07:08 a1.txt -rwxrwxr-x 1 ubuntu ubuntu 5 Sep 2 10:14 f1.txt -rwxrwxr-x 2 ubuntu ubuntu 20 Oct 8 07:08 file.txt -rwxrwxrwx 1 ubuntu ubuntu 134 Sep 23 09:14 s1.sh -rw-rw-r-- 1 ubuntu ubuntu 428 Oct 8 08:33 s10.sh -rwxrwxrwx 1 ubuntu ubuntu 198 Sep 23 09:23 s2.sh -rwxrwxr-x 1 ubuntu ubuntu 175 Oct 8 07:01 s3.sh -rwxrwxr-x 1 ubuntu ubuntu 96 Oct 8 07:10 s4.sh -rwxrwxr-x 1 ubuntu ubuntu 82 Oct 8 07:12 s5.sh -rwxrwxr-x 1 ubuntu ubuntu 149 Oct 8 07:25 s6.sh -rw-rw-r-- 1 ubuntu ubuntu 522 Oct 8 08:16 s7.sh -rw----- 1 ubuntu ubuntu 16 Oct 8 07:29 s7.sh.save -rw-rw-r-- 1 ubuntu ubuntu 105 Oct 8 08:22 s8.sh -rw-rw-r-- 1 ubuntu ubuntu 143 Oct 8 08:27 s9.sh</pre>

Que.	5. Update the existing file.
Command	gedit a1.txt
Output	<pre>ubuntu@ubuntu:~/Documents\$ gedit a1.txt</pre>

Que.	6. Apply ls -l and check the size of linked file created.
Command	ls -l
Output	<pre>ubuntu@ubuntu:~/Documents\$ ls -l total 68 drwxrwxr-x 2 ubuntu ubuntu 60 Aug 12 11:05 ABC -rwxrwxrwx 1 ubuntu ubuntu 102 Sep 2 10:48 S1.sh -rwxrwxr-x 1 ubuntu ubuntu 14 Oct 8 07:08 S1.txt -rwxrwxr-x 1 ubuntu ubuntu 29 Sep 2 09:59 S2.sh -rwxrwxr-x 2 ubuntu ubuntu 44 Oct 8 08:40 a1.txt -rwxrwxr-x 1 ubuntu ubuntu 5 Sep 2 10:14 f1.txt -rwxrwxr-x 2 ubuntu ubuntu 44 Oct 8 08:40 file.txt -rwxrwxrwx 1 ubuntu ubuntu 134 Sep 23 09:14 s1.sh -rw-rw-r-- 1 ubuntu ubuntu 428 Oct 8 08:33 s10.sh -rwxrwxrwx 1 ubuntu ubuntu 198 Sep 23 09:23 s2.sh -rwxrwxr-x 1 ubuntu ubuntu 175 Oct 8 07:01 s3.sh -rwxrwxr-x 1 ubuntu ubuntu 96 Oct 8 07:10 s4.sh -rwxrwxr-x 1 ubuntu ubuntu 82 Oct 8 07:12 s5.sh -rwxrwxr-x 1 ubuntu ubuntu 149 Oct 8 07:25 s6.sh -rw-rw-r-- 1 ubuntu ubuntu 522 Oct 8 08:16 s7.sh -rw----- 1 ubuntu ubuntu 16 Oct 8 07:29 s7.sh.save -rw-rw-r-- 1 ubuntu ubuntu 105 Oct 8 08:22 s8.sh -rw-rw-r-- 1 ubuntu ubuntu 143 Oct 8 08:27 s9.sh</pre>

Que.	7. Check the content of both the files and write your observation.
Command	cat a1.txt cat file.txt
Output	<pre>ubuntu@ubuntu:~/Documents\$ cat a1.txt Hello, How are you? this is updated content ubuntu@ubuntu:~/Documents\$ cat file.txt Hello, How are you? this is updated content</pre>

Que.	8. Delete the existing file on which you have created the link.
Command	rm a1.txt
Output	<pre>ubuntu@ubuntu:~/Documents\$ rm a1.txt</pre>

Que.	9. Apply ls -l and observe the output.
Command	ls -l
Output	<pre>ubuntu@ubuntu:~/Documents\$ ls -l total 64 drwxrwxr-x 2 ubuntu ubuntu 60 Aug 12 11:05 ABC -rwxrwxrwx 1 ubuntu ubuntu 102 Sep 2 10:48 S1.sh -rwxrwxr-x 1 ubuntu ubuntu 14 Oct 8 07:08 S1.txt -rwxrwxr-x 1 ubuntu ubuntu 29 Sep 2 09:59 S2.sh -rwxrwxr-x 1 ubuntu ubuntu 5 Sep 2 10:14 f1.txt -rwxrwxr-x 1 ubuntu ubuntu 44 Oct 8 08:40 file.txt -rwxrwxrwx 1 ubuntu ubuntu 134 Sep 23 09:14 s1.sh -rw-rw-r-- 1 ubuntu ubuntu 428 Oct 8 08:33 s10.sh -rwxrwxrwx 1 ubuntu ubuntu 198 Sep 23 09:23 s2.sh -rwxrwxr-x 1 ubuntu ubuntu 175 Oct 8 07:01 s3.sh -rwxrwxr-x 1 ubuntu ubuntu 96 Oct 8 07:10 s4.sh -rwxrwxr-x 1 ubuntu ubuntu 82 Oct 8 07:12 s5.sh -rwxrwxr-x 1 ubuntu ubuntu 149 Oct 8 07:25 s6.sh -rw-rw-r-- 1 ubuntu ubuntu 522 Oct 8 08:16 s7.sh -rw----- 1 ubuntu ubuntu 16 Oct 8 07:29 s7.sh.save -rw-rw-r-- 1 ubuntu ubuntu 105 Oct 8 08:22 s8.sh -rw-rw-r-- 1 ubuntu ubuntu 143 Oct 8 08:27 s9.sh</pre>

Que.	10. Perform exercise 2 to 9 for creation of soft link and write your observation. (ln -s [original file] [link name])
Command	ln -s a1.txt file2.txt ls -l
Output	<pre>ubuntu@ubuntu:~/Documents\$ ln -s a1.txt file2.txt ubuntu@ubuntu:~/Documents\$ ls -l total 64 drwxrwxr-x 2 ubuntu ubuntu 60 Aug 12 11:05 ABC -rwxrwxrwx 1 ubuntu ubuntu 102 Sep 2 10:48 S1.sh -rwxrwxr-x 1 ubuntu ubuntu 14 Oct 8 07:08 S1.txt -rwxrwxr-x 1 ubuntu ubuntu 29 Sep 2 09:59 S2.sh -rwxrwxr-x 1 ubuntu ubuntu 5 Sep 2 10:14 f1.txt -rwxrwxr-x 1 ubuntu ubuntu 44 Oct 8 08:40 file.txt lrwxrwxrwx 1 ubuntu ubuntu 6 Oct 8 08:44 file2.txt -> a1.txt -rwxrwxrwx 1 ubuntu ubuntu 134 Sep 23 09:14 s1.sh -rw-rw-r-- 1 ubuntu ubuntu 428 Oct 8 08:33 s10.sh -rwxrwxrwx 1 ubuntu ubuntu 198 Sep 23 09:23 s2.sh -rwxrwxr-x 1 ubuntu ubuntu 175 Oct 8 07:01 s3.sh -rwxrwxr-x 1 ubuntu ubuntu 96 Oct 8 07:10 s4.sh -rwxrwxr-x 1 ubuntu ubuntu 82 Oct 8 07:12 s5.sh -rwxrwxr-x 1 ubuntu ubuntu 149 Oct 8 07:25 s6.sh -rw-rw-r-- 1 ubuntu ubuntu 522 Oct 8 08:16 s7.sh -rw----- 1 ubuntu ubuntu 16 Oct 8 07:29 s7.sh.save -rw-rw-r-- 1 ubuntu ubuntu 105 Oct 8 08:22 s8.sh -rw-rw-r-- 1 ubuntu ubuntu 143 Oct 8 08:27 s9.sh</pre>

Que.	11. Write difference between hard link and soft link.
Output	<p>Hard Link:</p> <ul style="list-style-type: none"> • Points directly to the inode of a file. • Both the original file and hard link share the same inode. • Deleting the original file does not remove the hard link; data remains accessible. <p>Soft Link (Symbolic Link):</p> <ul style="list-style-type: none"> • Acts as a shortcut to another file. • Points to the file's path, not the inode. <p>If the original file is deleted, the soft link becomes broken.</p>

Que.	12. Apply ls -l /dev/sda1
Command	ls -l /dev/sda1
Output	<pre>ubuntu@ubuntu:~/Documents\$ ls -l /dev/sda brw-rw---- 1 root disk 8, 0 Aug 12 09:33 /dev/sda</pre>

Que.	13. To get an overview of local and remote file system devices and the amount of free space available, run the df command
Command	ls -df
Output	<pre>ubuntu@ubuntu:~/Documents\$ ls -df .</pre>

Que.	14. Apply df -h and see the difference in the output
Command	df -h
Output	<pre>ubuntu@ubuntu:~/Documents\$ df -h Filesystem Size Used Avail Use% Mounted on tmpfs 197M 1.9M 195M 1% /run /dev/sr0 5.7G 5.7G 0 100% /cdrom /cow 984M 312M 672M 32% / tmpfs 984M 8.0K 984M 1% /dev/shm tmpfs 5.0M 8.0K 5.0M 1% /run/lock tmpfs 984M 600K 984M 1% /tmp tmpfs 197M 160K 197M 1% /run/user/1000</pre>

Que.	15. For more detailed information about space used by a certain directory tree, use the du command. Apply du /home/ID_No
Command	du
Output	<pre>ubuntu@ubuntu:~/Documents\$ du 0 ./ABC 64 .</pre>

Que.	16. Apply du -h /home/ID_No and see the difference in the output
Command	Du -h /home/ubuntu
Output	<pre>ubuntu@ubuntu:~/Documents\$ du -h /home/ubuntu 4.0K /home/ubuntu/Desktop 0 /home/ubuntu/.cache/gvfsd 4.0K /home/ubuntu/.cache/mesa_shader_cache/25 8.0K /home/ubuntu/.cache/mesa_shader_cache/3a 8.0K /home/ubuntu/.cache/mesa_shader_cache/d2 12K /home/ubuntu/.cache/mesa_shader_cache/d4 4.0K /home/ubuntu/.cache/mesa_shader_cache/39 12K /home/ubuntu/.cache/mesa_shader_cache/ed 16K /home/ubuntu/.cache/mesa_shader_cache/a2 12K /home/ubuntu/.cache/mesa_shader_cache/33 4.0K /home/ubuntu/.cache/mesa_shader_cache/29 4.0K /home/ubuntu/.cache/mesa_shader_cache/28 8.0K /home/ubuntu/.cache/mesa_shader_cache/34 12K /home/ubuntu/.cache/mesa_shader_cache/a9 8.0K /home/ubuntu/.cache/mesa_shader_cache/1b 4.0K /home/ubuntu/.cache/mesa_shader_cache/5d 4.0K /home/ubuntu/.cache/mesa_shader_cache/b3 8.0K /home/ubuntu/.cache/mesa_shader_cache/f2 12K /home/ubuntu/.cache/mesa_shader_cache/09 8.0K /home/ubuntu/.cache/mesa_shader_cache/c5 12K /home/ubuntu/.cache/mesa_shader_cache/fa 8.0K /home/ubuntu/.cache/mesa_shader_cache/57 8.0K /home/ubuntu/.cache/mesa_shader_cache/c3 4.0K /home/ubuntu/.cache/mesa_shader_cache/8c 8.0K /home/ubuntu/.cache/mesa_shader_cache/e1 20K /home/ubuntu/.cache/mesa_shader_cache/75 8.0K /home/ubuntu/.cache/mesa_shader_cache/e7</pre>

Que.	17. Use the lsblk command to list the details of a specified block device or all the available devices
Command	lsblk
Output	<pre>ubuntu@ubuntu:~/Documents\$ lsblk NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS loop0 7:0 0 1.6G 1 loop /rofs loop1 7:1 0 457.5M 1 loop loop2 7:2 0 868.1M 1 loop loop3 7:3 0 10.7M 1 loop /snap/firmware-updater/127 loop4 7:4 0 269.6M 1 loop /snap/firefox/4173 loop5 7:5 0 505.1M 1 loop /snap/gnome-42-2204/176 loop6 7:6 0 38.7M 1 loop /snap/snapd/21465 loop7 7:7 0 74.2M 1 loop /snap/core22/1380 loop8 7:8 0 4K 1 loop /snap/bare/5 loop9 7:9 0 476K 1 loop /snap/snapd-desktop-integration/157 loop10 7:10 0 10.3M 1 loop /snap/snap-store/1124 loop11 7:11 0 91.7M 1 loop /snap/gtk-common-themes/1535 loop12 7:12 0 137.3M 1 loop /snap/thunderbird/470 loop13 7:13 0 116.7M 1 loop /snap/ubuntu-desktop-bootstrap/171 sda 8:0 0 25G 0 disk sr0 11:0 1 5.7G 0 rom /cdrom</pre>

Que.	18. Apply locate passwd and see the output.
Command	locate passwd
Output	<pre>ubuntu@ubuntu:~/Documents\$ locate passwd /etc/passwd /etc/passwd- /etc/pam.d/chpasswd /etc/pam.d/passwd /etc/security/opasswd /rofs/etc/passwd /rofs/etc/pam.d/chpasswd /rofs/etc/pam.d/passwd /rofs/etc/security/opasswd /rofs/usr/bin/gpasswd /rofs/usr/bin/grub-nkpasswd-pbkdf2 /rofs/usr/bin/ldappasswd /rofs/usr/bin/passwd /rofs/usr/lib/python3/dist-packages/cloudinit/sources/helpers/vmware/imc/config_passwd.py /rofs/usr/lib/python3/dist-packages/cloudinit/sources/helpers/vmware/_pycache__config_passwd.cpython-312.pyc /rofs/usr/lib/tmpfiles.d/passwd.conf /rofs/usr/lib/x86_64-linux-gnu/samba/libsmbpasswddparser-samba4.so.0 /rofs/usr/sbin/chpasswd /rofs/usr/sbin/chgpasswd /rofs/usr/share/base-passwd /rofs/usr/share/base-passwd/group.master /rofs/usr/share/base-passwd/passwd.master /rofs/usr/share/bash-completion/completions/chpasswd /rofs/usr/share/bash-completion/completions/gpasswd /rofs/usr/share/bash-completion/completions/grub-nkpasswd-pbkdf2</pre>
Que.	19. To search for files by file name, use the -name FILENAME option. With this option, find returns the path to files matching FILENAME exactly. Search for files named sshd_config starting from the / directory. (find / -name sshd_config)
Command	find / -name sshd_config
Output	<pre>ubuntu@ubuntu:~/Documents\$ find / -name sshd_config find: '/etc/credstore': Permission denied find: '/etc/credstore.encrypted': Permission denied find: '/etc/cups/ssl': Permission denied find: '/etc/polkit-1/rules.d': Permission denied /etc/ssh/sshd_config find: '/etc/ssl/private': Permission denied find: '/etc/sssd': Permission denied find: '/etc/sudoers.d': Permission denied find: '/home/installer': Permission denied find: '/home/shiv': Permission denied find: '/proc/tty/driver': Permission denied find: '/proc/1/task/1/fd': Permission denied find: '/proc/1/task/1/fdinfo': Permission denied find: '/proc/1/task/1/ns': Permission denied find: '/proc/1/fd': Permission denied find: '/proc/1/map_files': Permission denied find: '/proc/1/fdinfo': Permission denied find: '/proc/1/ns': Permission denied find: '/proc/2/task/2/fd': Permission denied find: '/proc/2/task/2/fdinfo': Permission denied find: '/proc/2/task/2/ns': Permission denied find: '/proc/2/fd': Permission denied find: '/proc/2/map_files': Permission denied find: '/proc/2/fdinfo': Permission denied</pre>

Que.	20. search for files starting in the / directory that end in .txt.
Command	find / -name .txt
Output	<pre>ubuntu@ubuntu:~/Documents\$ find / -name .txt find: '/etc/credstore': Permission denied find: '/etc/credstore.encrypted': Permission denied find: '/etc/cups/ssl': Permission denied find: '/etc/polkit-1/rules.d': Permission denied find: '/etc/ssl/private': Permission denied find: '/etc/sssd': Permission denied find: '/etc/sudoers.d': Permission denied find: '/home/installer': Permission denied find: '/home/shiv': Permission denied find: '/proc/tty/driver': Permission denied find: '/proc/1/task/1/fd': Permission denied find: '/proc/1/task/1/fdinfo': Permission denied find: '/proc/1/task/1/ns': Permission denied find: '/proc/1/fd': Permission denied find: '/proc/1/map_files': Permission denied find: '/proc/1/fdinfo': Permission denied find: '/proc/1/ns': Permission denied find: '/proc/2/task/2/fd': Permission denied find: '/proc/2/task/2/fdinfo': Permission denied find: '/proc/2/task/2/ns': Permission denied find: '/proc/2/fd': Permission denied find: '/proc/2/map_files': Permission denied find: '/proc/2/fdinfo': Permission denied</pre>

Que.	21. search for files in the /etc/ directory that contain the word, “pass”.
Command	find /etc -name "pass*"
Output	<pre>ubuntu@ubuntu:~/Documents\$ find /etc -name "pass*" find: '/etc/credstore': Permission denied find: '/etc/credstore.encrypted': Permission denied find: '/etc/cups/ssl': Permission denied /etc/pam.d/passwd /etc/passwd find: '/etc/polkit-1/rules.d': Permission denied find: '/etc/ssl/private': Permission denied find: '/etc/sssd': Permission denied find: '/etc/sudoers.d': Permission denied /etc/passwd-</pre>

Que.	22. Search for files owned by user in the /home/ID_No directory.
Command	find /home/ubuntu -user ubuntu
Output	<pre>ubuntu@ubuntu:~/Documents\$ find /home/ubuntu -user ubuntu /home/ubuntu /home/ubuntu/.bash_logout /home/ubuntu/.bashrc /home/ubuntu/.profile /home/ubuntu/Desktop /home/ubuntu/Desktop/ubuntu-desktop-bootstrap_ubuntu-desktop-bootstrap.desktop /home/ubuntu/.cache /home/ubuntu/.cache/gvfsd /home/ubuntu/.cache/mesa_shader_cache /home/ubuntu/.cache/mesa_shader_cache/index /home/ubuntu/.cache/mesa_shader_cache/25 /home/ubuntu/.cache/mesa_shader_cache/25/2d9694e78cfaca3ead7af12eb7e2173024d311 /home/ubuntu/.cache/mesa_shader_cache/3a /home/ubuntu/.cache/mesa_shader_cache/3a/83eab0826c783f04438a47a48d7df9000d319f /home/ubuntu/.cache/mesa_shader_cache/3a/39935a47c5063dddefcdd648a8c82be1a309cf /home/ubuntu/.cache/mesa_shader_cache/d2 /home/ubuntu/.cache/mesa_shader_cache/d2/05aa6bc6b6dbfd566e27ac6776093ca8c06615 /home/ubuntu/.cache/mesa_shader_cache/d2/f84cd4f8464be9faa1181cb16053ebfe3339b6 /home/ubuntu/.cache/mesa_shader_cache/d4 /home/ubuntu/.cache/mesa_shader_cache/d4/56633fc70475c599e2c8ba0ce2f524bb3de52f /home/ubuntu/.cache/mesa_shader_cache/d4/2619b394f8146a7680c09e6cbd44f7b114d6fe /home/ubuntu/.cache/mesa_shader_cache/39 /home/ubuntu/.cache/mesa_shader_cache/39/e2d57e48383a2fc67a7a1f54e386e341124cde /home/ubuntu/.cache/mesa_shader_cache/ed</pre>

Que.	23. Search for files owned by the group user in the /home/ID_No.
Command	find /home/ubuntu -group ubuntu
Output	<pre>ubuntu@ubuntu:~/Documents\$ find /home/ubuntu -group ubuntu /home/ubuntu /home/ubuntu/.bash_logout /home/ubuntu/.bashrc /home/ubuntu/.profile /home/ubuntu/Desktop /home/ubuntu/Desktop/ubuntu-desktop-bootstrap_ubuntu-desktop-bootstrap.desktop /home/ubuntu/.cache /home/ubuntu/.cache/gvfsd /home/ubuntu/.cache/mesa_shader_cache /home/ubuntu/.cache/mesa_shader_cache/index /home/ubuntu/.cache/mesa_shader_cache/25 /home/ubuntu/.cache/mesa_shader_cache/25/2d9694e78cfaca3ead7af12eb7e2173024d311 /home/ubuntu/.cache/mesa_shader_cache/3a /home/ubuntu/.cache/mesa_shader_cache/3a/83eab0826c783f04438a47a48d7df9000d319f /home/ubuntu/.cache/mesa_shader_cache/3a/39935a47c5063dddefcdd648a8c82be1a309cf /home/ubuntu/.cache/mesa_shader_cache/d2 /home/ubuntu/.cache/mesa_shader_cache/d2/05aa6bc6b6dbfd566e27ac6776093ca8c06615 /home/ubuntu/.cache/mesa_shader_cache/d2/f84cd4f8464be9faa1181cb16053ebfe3339b6 /home/ubuntu/.cache/mesa_shader_cache/d4 /home/ubuntu/.cache/mesa_shader_cache/d4/56633fc70475c599e2c8ba0ce2f524bb3de52f /home/ubuntu/.cache/mesa_shader_cache/d4/2619b394f8146a7680c09e6cbd44f7b114d6fe /home/ubuntu/.cache/mesa_shader_cache/39 /home/ubuntu/.cache/mesa_shader_cache/39/e2d57e48383a2fc67a7a1f54e386e341124cde /home/ubuntu/.cache/mesa_shader_cache/ed /home/ubuntu/.cache/mesa_shader_cache/ed/3dabd75c5a8e8cf35b7070ac279fbe4f625056 /home/ubuntu/.cache/mesa_shader_cache/ed/723cf1e5edb524d6f741e20b60d42668ac9b2a /home/ubuntu/.cache/mesa_shader_cache/ed/2ca6dd632e6e2aa5110ee03f7e4ca6b7563fe1</pre>

Que.	24. Apply df -h to see the partitions.
Command	df -h
Output	<pre>ubuntu@ubuntu:~/Documents\$ df -h Filesystem Size Used Avail Use% Mounted on tmpfs 197M 1.9M 195M 1% /run /dev/sr0 5.7G 5.7G 0 100% /cdrom /cow 984M 321M 664M 33% / tmpfs 984M 8.0K 984M 1% /dev/shm tmpfs 5.0M 8.0K 5.0M 1% /run/lock tmpfs 984M 776K 983M 1% /tmp tmpfs 197M 160K 197M 1% /run/user/1000</pre>

Que.	25. Unmount your USB drive by locating its path
Command	sudo umount /dev/sdX1
Output	<pre>ubuntu@ubuntu:~/Documents\$ sudo umount /dev/sdX1 umount: /dev/sdX1: no mount point specified.</pre>

Que.	26. Again, apply df -h and check your USB if it is accessible or not.
Command	df -h
Output	<pre>ubuntu@ubuntu:~/Documents\$ df -h Filesystem Size Used Avail Use% Mounted on tmpfs 197M 1.9M 195M 1% /run /dev/sr0 5.7G 5.7G 0 100% /cdrom /cow 984M 321M 664M 33% / tmpfs 984M 8.0K 984M 1% /dev/shm tmpfs 5.0M 8.0K 5.0M 1% /run/lock tmpfs 984M 776K 983M 1% /tmp tmpfs 197M 160K 197M 1% /run/user/1000</pre>

Que.	27. Mount your USB drive into directory. (Create one directory and inside that apply mount drive directory name command)
Command	mkdir ~/usb_mount sudo umount /dev/sdX1 ~/usb_mount
Output	<pre>ubuntu@ubuntu:~/Documents\$ mkdir ~/usb_mount ubuntu@ubuntu:~/Documents\$ sudo umount /dev/sdX1 ~/usb_mount umount: /dev/sdX1: no mount point specified. umount: /home/ubuntu/usb_mount: not mounted.</pre>

Que.	28. Again, apply df -h command and check whether it is mounted or not.
Command	df -h
Output	<pre>ubuntu@ubuntu:~/Documents\$ df -h Filesystem Size Used Avail Use% Mounted on tmpfs 197M 1.9M 195M 1% /run /dev/sr0 5.7G 5.7G 0 100% /cdrom /cow 984M 321M 664M 33% / tmpfs 984M 8.0K 984M 1% /dev/shm tmpfs 5.0M 8.0K 5.0M 1% /run/lock tmpfs 984M 776K 983M 1% /tmp tmpfs 197M 160K 197M 1% /run/user/1000</pre>

Practical 5

Aim: Shell scripting

Exercise:

Que.	1. Check whether the given file exists or not.
Command	<pre>#!/bin/bash read -p "Enter the filename: " filename if [-f "\$filename"]; then echo "\$filename exists" else echo "\$filename doesn't exist" fi</pre>
Output	<pre>ubuntu@ubuntu:~/Documents\$ bash s1.sh Enter Filename S1.txt File exists</pre>

Que.	2. Check whether the argument passed from command line is file or directory.
Command	<pre>if test -f "\$1"; then echo "\$1 is a file" elif test -d "\$1"; then echo "\$1 is a directory" else echo "\$1 is neither file nor directory" fi</pre>
Output	<pre>ubuntu@ubuntu:~/Documents\$ bash s2.sh s2.sh: line 1: !#/bin/bash: No such file or directory Enter File/Directory name S1.txt 'S1.txt' is a File ubuntu@ubuntu:~/Documents\$ bash s2.sh s2.sh: line 1: !#/bin/bash: No such file or directory Enter File/Directory name ABC 'ABC' is a Directory</pre>

Que.	3. List out all empty files in current working directory. Directory may contain subdirectories also.
Command	<pre>#!/bin/bash check() { for i in "\$1"/* do if [-f "\$i"]; then if [! -s "\$i"]; then echo "Empty file: \$i" fi elif [-d "\$i"]; then check "\$i" fi done } check .</pre>
Output	<pre>ubuntu@ubuntu:~/Documents\$ bash s3.sh Empty file : ./S1.txt Empty file : ./a1.txt</pre>

Que.	4. Give two file names as command line arguments and check both the files are same or different. If they are same then delete the second file otherwise suggest what changes are required to make 1st file similar to second file.
Command	<pre>if cmp -s "\$1" "\$2"; then echo "both files are same" else echo "different" diff "\$1" "\$2" fi</pre>
Output	<pre>ubuntu@ubuntu:~/Documents\$ bash s4.sh S1.txt a1.txt different 1c1 < Hello World! --- > Hello, How are you?</pre>

Que.	5. Print multiplication table of given number
Command	<pre>echo "Enter a number: " read n for i in {1..10} do echo "\$n * \$i = \$((n * i))" done</pre>
Output	<pre>ubuntu@ubuntu:~/Documents\$ bash s5.sh Enter a Number 19 19 * 1 = 19 19 * 2 = 38 19 * 3 = 57 19 * 4 = 76 19 * 5 = 95 19 * 6 = 114 19 * 7 = 133 19 * 8 = 152 19 * 9 = 171 19 * 10 = 190</pre>

Que.	6. Shell script to check executable rights for all files in the current directory, if a file does not have the execute permission then make it executable.
Command	<pre>for i in * do if test -f "\$i"; then if test -x "\$i"; then echo "Executable" else echo "Making executable" chmod +x "\$i" fi fi done</pre>
Output	<pre>ubuntu@ubuntu:~/Documents\$ bash s6.sh Executable Making executable Making executable Making executable Making executable Executable Executable Making executable Making executable Making executable Making executable</pre>

Que.	7. Write a shell script for arithmetic calculator using command line arguments.
Command	<pre> if ["\$#" -ne 3]; then echo "Usage: \$0 <number1> <operator> <number2>" echo "Operators: +, -, *, /" exit 1 fi num1=\$1 operator=\$2 num2=\$3 case \$operator in +) result=\$(bc <<< "\$num1 + \$num2") ;; -) result=\$(bc <<< "\$num1 - \$num2") ;; *) result=\$(bc <<< "\$num1 * \$num2") ;; '/') if ["\$num2" -eq 0]; then echo "Error: Division by zero is not allowed." exit 2 fi result=\$(echo "scale=2; \$num1 / \$num2" bc) ;; *) echo "Error: Invalid operator, Use +, -, *, or /." exit 1 ;; esac echo "Result: \$result" </pre>

Output

```
ubuntu@ubuntu:~/Documents$ bash s7.sh 5 + 5
Result: 10
ubuntu@ubuntu:~/Documents$ bash s7.sh 10 - 5
Result: 5
ubuntu@ubuntu:~/Documents$ bash s7.sh 10 \* 5
Result: 50
ubuntu@ubuntu:~/Documents$ bash s7.sh 10 / 5
Result: 50
```

Que.**8. Write a script to print a given number in reversed order.****Command**

```
read -p "Enter a number: " n

while [ $n -gt 0 ]
do
    p=$((n % 10))
    echo -n "$p"
    n=$((n / 10))
done

echo
```

Output

```
ubuntu@ubuntu:~/Documents$ bash s8.sh
Enter a number: 14568
86541
```

Que.**9. Write a script to convert string from lower to upper and upper to lower case.****Command**

```
read -p "Enter a string: " s

l=$(tr '[:upper:]' '[:lower:]' <<< "$s")
u=$(tr '[:lower:]' '[:upper:]' <<< "$s")

echo "$s"
echo "$l"
echo "$u"
```

Output

```
ubuntu@ubuntu:~/Documents$ bash s9.sh
Enter a String: OperatingSystem
OperatingSystem
operatingsystem
OPERATINGSYSTEM
```

Que.**10. Shell script to Create a menu as shown below using the case statement**

- 1) list of files
- 2) today's date
- 3) users of system

	4) processes of user 5) exit to prompt
Command	<pre> while true; do echo "1. List of files" echo "2. Today's date" echo "3. Users of system" echo "4. Process of user" echo "5. Exit to prompt" read -p "Enter your choice: " n case \$n in 1) echo "List of files:" ls ;; 2) echo "Today's date:" date ;; 3) echo "Users of system:" who ;; 4) echo "Processes of current user:" ps -u \$(whoami) ;; 5) echo "Exit to prompt" exit 0 ;; *) echo "Invalid option" ;; esac done </pre>

Output

```
ubuntu@ubuntu:~/Documents$ bash s10.sh
1. List of files
2. Today's date
3. USers of system
4. Process of user
5. Exit to prompt
Enter your choice: 1
List of files:
ABC      S1.txt  a1.txt  s1.sh   s2.sh  s4.sh  s6.sh  s7.sh.save  s9.sh
S1.sh    S2.sh   f1.txt  s10.sh  s3.sh  s5.sh  s7.sh  s8.sh
1. List of files
2. Today's date
3. USers of system
4. Process of user
5. Exit to prompt
Enter your choice: 2
Today's date:3
Tue Oct  8 08:33:22 UTC 2024
1. List of files
2. Today's date
3. USers of system
4. Process of user
5. Exit to prompt
Enter your choice: 3
Users of System:
ubuntu  seat0          2024-08-12 09:32 (login screen)
ubuntu  :0              2024-08-12 09:32 (:0)
```

```
1. List of files
2. Today's date
3. USers of system
4. Process of user
5. Exit to prompt
Enter your choice: 5
Exit to prompt
```

```
ubuntu@ubuntu:~/Documents$ █
```

```
Enter your choice: 3
Users of System:
ubuntu    seat0      2024-08-12 09:32 (login screen)
ubuntu    :0        2024-08-12 09:32 (:0)
1. List of files
2. Today's date
3. USers of system
4. Process of user
5. Exit to prompt
Enter your choice: 4
Processes of current user:
  PID TTY          TIME CMD
 1659 ?      00:00:07 systemd
 1660 ?      00:00:00 (sd-pam)
 1677 ?      00:00:14 pipewire
 1678 ?      00:00:00 pipewire
 1680 ?      00:00:09 wireplumber
 1681 ?      00:00:06 pipewire-pulse
 1682 ?      00:00:01 gnome-keyring-d
 1687 ?      00:00:13 dbus-daemon
 1720 tty2    00:00:00 gdm-x-session
 1724 tty2    00:13:06 Xorg
 1923 tty2    00:00:00 gnome-session-b
 2112 ?      00:00:00 at-spi-bus-laun
 2119 ?      00:00:01 dbus-daemon
 2170 ?      00:00:00 gcr-ssh-agent
 2171 ?      00:00:00 gnome-session-c
 2187 ?      00:00:00 gvfsd
 2199 ?      00:00:00 gvfsd-fuse
 2225 ?      00:00:00 gnome-session-b
 2254 ?      00:18:42 gnome-shell
```

Practical 6

Aim: Process control system calls:

Que.	The demonstration of fork () system call. Write a program which takes number of processes from the user and create those processes. (The fork system call will create processes in power of 2.)
Code	<pre>#include <stdio.h> #include <stdlib.h> #include <sys/types.h> #include <sys/wait.h> #include <unistd.h> int main(void) { pid_t pid = fork(); if (pid == 0) { printf("Child ==> PPID: %d PID: %d\n", getppid(), getpid()); } else if (pid > 0) { printf("Parent => PID: %d\n", getpid()); printf("Waiting for child process to finish.\n"); wait(NULL); printf("Child process finished.\n"); } else { printf("Unable to create child process.\n"); } return EXIT_SUCCESS; }</pre>
Output	<pre>ubuntu@ubuntu:~/Documents/Practical6\$ gcc -o Prac6_1 Prac6_1.c ubuntu@ubuntu:~/Documents/Practical6\$./Prac6_1 Parent => PID: 14636 Waiting for child process to finish. Child ==> PPID: 14636 PID: 14637 Child process finished.</pre>

Que.	The demonstration of fork () system call. Write a program which takes number of processes from the user and create those processes. (The fork system call will create processes in power of 2.)
Code	<pre>#include <stdio.h> #include <sys/types.h> #include <unistd.h> int main() { fork(); printf("Hello\n"); return 0; }</pre>
Output	<pre>ubuntu@ubuntu:~/Documents/Practical6\$ gcc -o Prac6_2 Prac6_2.c ubuntu@ubuntu:~/Documents/Practical6\$./Prac6_2 Hello Hello</pre>

Que.	The demonstration of fork () system call. Write a program which takes number of processes from the user and create those processes. (The fork system call will create processes in power of 2.)
Code	<pre>#include <stdio.h> #include <sys/types.h> #include <unistd.h> int main() { fork(); fork(); fork(); printf("Hello\n"); return 0; }</pre>
Output	<pre>ubuntu@ubuntu:~/Documents/Practical6\$ gcc -o Prac6_3 Prac6_3.c ubuntu@ubuntu:~/Documents/Practical6\$./Prac6_3 Hello Hello Hello ubuntu@ubuntu:~/Documents/Practical6\$ Hello Hello Hello Hello Hello</pre>

Que.	The demonstration of fork () system call. Write a program which takes number of processes from the user and create those processes. (The fork system call will create processes in power of 2.)
Code	<pre>#include <stdio.h> #include <sys/types.h> #include <unistd.h> void forkexample() { if (fork() == 0) { printf("Child!\n"); } else { printf("Parent!\n"); } } int main() { forkexample(); return 0; }</pre>
Output	<pre>ubuntu@ubuntu:~/Documents/Practical6\$ gcc -o Prac6_4 Prac6_4.c ubuntu@ubuntu:~/Documents/Practical6\$./Prac6_4 Parent! Child!</pre>

Summary:

In this practical exercise, the goal was to demonstrate the use of the fork() system call in process creation and management. The specific aim was to create a user-defined number of processes, noting that the fork() system call inherently creates processes in powers of 2.

Summary of Exercises:

1. In this exercise, a simple program is created using the fork() system call to demonstrate the creation of a child process from a parent process. The child process prints its process ID (PID) and the parent's PID (PPID), while the parent process waits for the child to complete before exiting. This shows the synchronization between parent and child processes using wait().
2. This exercise involved a program that uses fork() to create a new process, which then prints "Hello world!" from both the parent and the child process. The program demonstrates that fork() creates an exact copy of the calling process, leading to multiple outputs due to multiple process instances.
3. In this example, three consecutive calls to fork() are made, resulting in the creation of multiple processes. The number of processes created is in the power of 2, specifically $2^3=8$. Each process prints "Hello", demonstrating the exponential growth in the number of processes created when fork() is called multiple times.

4. Here, the fork() system call is encapsulated within a function. Depending on whether fork() returns 0 (child process) or a non-zero value (parent process), the function prints either "Child!" or "Parent!". This example shows how fork() can be used within a function to manage process creation and differentiate behavior between parent and child processes.

Conclusion:

The practical session illustrated the mechanics of the fork() system call and its impact on process creation. By experimenting with different uses of fork(), such as creating multiple processes, synchronizing parent and child processes, and managing process behaviors, you gained a deeper understanding of how process control works in UNIX-like operating systems. The exercises demonstrated that each call to fork() results in a new child process, with the total number of processes doubling with each call, hence following the power of 2 growth pattern.



Practical 7

Aim: Demonstration of execve () and wait () system calls along with zombie and orphan states.

Que.	1. Zombie Process
Code	#include <stdlib.h> #include <sys/types.h> #include <unistd.h> int main() { pid_t child_pid = fork(); if (child_pid > 0) { sleep(50); } else { exit(0); } return 0; }
Output	<code>ubuntu@ubuntu:~/Documents/Practical7\$ g++ Prac7_1.c -o Prac7_1 ubuntu@ubuntu:~/Documents/Practical7\$./Prac7_1</code>

Que.	2. Orphan Process
Code	<pre>#include <stdio.h> #include <sys/types.h> #include <unistd.h> int main() { int pid = fork(); if (pid > 0) { printf("in parent process\n"); } else if (pid == 0) { sleep(30); printf("in child process\n"); } return 0; }</pre>
Output	<pre>ubuntu@ubuntu:~/Documents/Practical7\$ g++ Prac7_2.c -o Prac7_2 ubuntu@ubuntu:~/Documents/Practical7\$./Prac7_2 in parent process ubuntu@ubuntu:~/Documents/Practical7\$ in child process</pre>

Que.	3. Execvparent Execvchild
Code	<p style="text-align: center;">Execvparent</p> <pre>#include <stdio.h> #include <unistd.h> #include <stdlib.h> int main(int argc, char *argv[]) { printf("\n-> PID of parent.c = %d ", getpid()); int parent = fork(); if (parent == -1) { printf("\n-> Some errors in calling"); } if (parent == 0) { printf("\n-> Now execv will call child.c from child process."); printf("\n-> The childe process in running."); char *args[] = {"239", NULL}; execv("./execvchild", args); } else {</pre>

```

        printf("\n-> Now parent is running \n");
    }

    return 0;
}

```

Execvchild

```

#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
    printf("\n-> Now we are in child.c");
    printf("\n-> The PID of child.c = %d", getpid());
    printf("\n-> %s", *argv);

    return 0;
}

```

Output

```

ubuntu@ubuntu:~/Documents/Practical7$ g++ Prac7_3_2.c -o Prac7_3_2
ubuntu@ubuntu:~/Documents/Practical7$ ./Prac7_3_2

-> Now we are in child.c
-> The PID of child.c = 16692
-> ./Prac7_3_2ubuntu@ubuntu:~/Documents/Practical7$ ^C

```

Summary:

The objective of Practical 7 was to demonstrate the use of the execv() and wait() system calls in the context of process control, and to illustrate the creation and handling of zombie and orphan processes.

Summary of Exercises:

1. Zombie Process:

- A zombie process is a process that has completed execution (terminated) but still has an entry in the process table. This happens when the parent process does not call wait() to read the child's exit status.
- In this exercise, the program forks to create a child process, which immediately terminates using exit(0), while the parent process sleeps for 50 seconds. During this sleep period, the child becomes a zombie because the parent does not call wait() to clear the terminated child's entry from the process table.

2. Orphan Process:

- An orphan process is a child process whose parent has terminated or exited. Orphan processes are adopted by the init process (PID 1) in UNIX-based systems.
- In this example, a child process is created using fork(). The parent process prints "in parent process" and then terminates. Meanwhile, the child process sleeps for 30 seconds before

printing "in child process". When the child process finally wakes up, its parent is no longer active, making it an orphan, and it is then adopted by the init process.

3. execv() System Call:

- The execv() system call replaces the current process image with a new process image. It is commonly used to run a different program within the same process.
- This exercise involves two files: execvparent.c and execvchild.c.
 - execvparent.c: The parent process is created, which in turn forks a child process. The child process uses the execv() call to replace its image with that of execvchild.c, effectively running the new program (execvchild.c).
 - execvchild.c: This program prints the PID of the current child process and some arguments passed by the parent. This exercise demonstrates how the execv() call transfers control to another program and how data can be passed between them.

Conclusion:

Practical 7 provided hands-on experience with advanced process control techniques in UNIX-like systems. The exercises helped demonstrate how to manage processes, replace them using the execv() system call, and handle special states like zombie and orphan processes. This understanding is fundamental in system programming, where process management is critical to ensuring resource efficiency and stability of applications.

Practical 8

Aim: Implementation of Process Scheduling Algorithm:

Que.	A. FCFS
Code	<pre>#include <iostream> #include <algorithm> using namespace std; typedef struct { int arrival_time, burst_time, waiting_time, pid, TAT; } Process; bool compare(Process a, Process b) { return a.arrival_time < b.arrival_time; } int main() { int N, total_waiting_time = 0, start_time = 0; float average_waiting_time = 0; cout << "Enter Number of Processes : "; cin >> N; cout << endl; Process process[N]; for(int i=0; i<N; i++) { cout << "Enter Burst time of P[" << (i+1) << "] : "; cin >> process[i].burst_time; cout << "Enter Arrival time of P[" << (i+1) << "] : "; cin >> process[i].arrival_time; process[i].pid = (i+1); cout << endl; } sort(process, process+N, compare); cout << "PID Arrival time Burst time Waiting Time TAT" << endl; for(int i=0; i<N; i++) { if(i == 0) { process[i].waiting_time = 0; start_time = process[i].arrival_time; } else process[i].waiting_time = start_time - process[i].arrival_time; start_time += process[i].burst_time; process[i].TAT = start_time - process[i].arrival_time; cout << process[i].pid << " " << process[i].arrival_time << " " << process[i].burst_time << " " << process[i].waiting_time << " " << process[i].TAT << endl; } }</pre>

```

        total_waiting_time += process[i].waiting_time;
        start_time += process[i].burst_time;
        process[i].TAT = start_time - process[i].arrival_time;
        cout << process[i].pid << "\t" << process[i].arrival_time << "\t\t" <<
        process[i].burst_time << "\t" << process[i].waiting_time << "\t" << process[i].TAT <<
        endl;
    }
    average_waiting_time = (float)total_waiting_time/N;
    cout << "\nTotal Waiting time : " << total_waiting_time << endl;
    cout << "Average Waiting time : " << average_waiting_time << endl;
return 0;
}

```

Output

```

ubuntu@ubuntu:~/Documents$ g++ Prac8_1.c -o Prac8_1
ubuntu@ubuntu:~/Documents$ ./Prac8_1
Enter Number of Processes : 5

Enter Burst time of P[1] : 6
Enter Arrival time of P[1] : 2

Enter Burst time of P[2] : 2
Enter Arrival time of P[2] : 5

Enter Burst time of P[3] : 8
Enter Arrival time of P[3] : 1

Enter Burst time of P[4] : 3
Enter Arrival time of P[4] : 0

Enter Burst time of P[5] : 4
Enter Arrival time of P[5] : 4

PID Arrival time Burst time Waiting Time TAT
4      0            3      0      3
3      1            8      2      10
1      2            6      9      15
5      4            4     13      17
2      5            2     16      18

Total Waiting time : 40
Average Waiting time : 8

```

Que.	B. Round Robing
Code	<pre>#include <iostream> #include <algorithm> using namespace std; typedef struct { int arrival_time, burst_time, waiting_time, pid, TAT, remaining_time, completion_time; bool completed; } Process; bool compareArrival(Process a, Process b) { return (a.arrival_time < b.arrival_time); } int main() { int N, quantum, completed = 0, total_waiting_time = 0, time = 0; float average_waiting_time = 0; cout << "Enter Number of Processes : "; cin >> N; cout << "Enter Quantum Time : "; cin >> quantum; cout << endl; Process process[N]; for(int i=0; i<N; i++) { cout << "Enter Burst time of P[" << (i+1) << "] : "; cin >> process[i].burst_time; cout << "Enter Arrival time of P[" << (i+1) << "] : "; cin >> process[i].arrival_time; process[i].pid = (i+1); process[i].remaining_time = process[i].burst_time; process[i].completed = false; cout << endl; } sort(process, process+N, compareArrival); cout << "PID Arrival time Burst time Waiting Time TAT Completion time" << endl; time += process[0].arrival_time; while(completed < N) { for(int i=0; i<N; i++) { if(process[i].completed == true) continue; else if(process[i].arrival_time > time) break; else { if(process[i].remaining_time > quantum) { time += quantum; process[i].remaining_time -= quantum; } else { time += process[i].remaining_time; process[i].remaining_time = 0; process[i].completed = true; } } } } }</pre>

```

process[i].completion_time = time;
process[i].TAT = process[i].completion_time - process[i].arrival_time;
process[i].waiting_time = process[i].TAT - process[i].burst_time;
total_waiting_time += process[i].waiting_time;
completed++;
cout << process[i].pid << "\t " << process[i].arrival_time << "\t\t" <<
process[i].burst_time << "\t " << process[i].waiting_time << "\t " << process[i].TAT <<
"\t\t" << process[i].completion_time << endl;
}
}
}
}
}

average_waiting_time = (float)total_waiting_time/N;
cout << "\nTotal Waiting time : " << total_waiting_time << endl;
cout << "Average Waiting time : " << average_waiting_time << endl;
return 0;
}

```

Output

```

ubuntu@ubuntu:~/Documents$ g++ Prac8_2.c -o Prac8_2
ubuntu@ubuntu:~/Documents$ ./Prac8_2
Enter Number of Processes : 3
Enter Quantum Time : 2

Enter Burst time of P[1] : 4
Enter Arrival time of P[1] : 0

Enter Burst time of P[2] : 3
Enter Arrival time of P[2] : 0

Enter Burst time of P[3] : 5
Enter Arrival time of P[3] : 0

PID Arrival time Burst time Waiting Time TAT Completion time
1      0            4        4      8          8
2      0            3        6      9          9
3      0            5        7     12         12

Total Waiting time : 17
Average Waiting time : 5.66667

```

Que.	C. SJF
Code	<pre>#include <iostream> #include <algorithm> using namespace std; typedef struct { int arrival_time, burst_time, waiting_time, pid, TAT; } Process; bool compareArrival(Process a, Process b) { return (a.arrival_time < b.arrival_time); } bool compareBurst(Process a, Process b) { return (a.burst_time < b.burst_time); } void sortByBurst(Process process[], int N); int main() { int N, total_waiting_time = 0, start_time = 0; float average_waiting_time = 0; cout << "Enter Number of Processes : "; cin >> N; cout << endl; Process process[N]; for(int i=0; i<N; i++) { cout << "Enter Burst time of P[" << (i+1) << "] : "; cin >> process[i].burst_time; cout << "Enter Arrival time of P[" << (i+1) << "] : "; cin >> process[i].arrival_time; process[i].pid = (i+1); cout << endl; } sort(process, process+N, compareArrival); sortByBurst(process, N); cout << "PID Arrival time Burst time Waiting Time TAT" << endl; start_time = process[0].arrival_time; for(int i=0; i<N; i++) { if(i == 0) process[i].waiting_time = 0; else process[i].waiting_time = start_time - process[i].arrival_time; total_waiting_time += process[i].waiting_time; start_time += process[i].burst_time; process[i].TAT = start_time - process[i].arrival_time; cout << process[i].pid << "\t " << process[i].arrival_time << "\t\t" << process[i].burst_time << "\t " << process[i].waiting_time << "\t " << process[i].TAT << endl; } average_waiting_time = (float)total_waiting_time/N; }</pre>

```

cout << "\nTotal Waiting time : " << total_waiting_time << endl;
cout << "Average Waiting time : " << average_waiting_time << endl;
return 0;
}
void sortByBurst(Process process[], int N) {
    int start_time = process[0].arrival_time;
    int j;
    for(int i=0; i<N-1; i++) {
        for(j=i+1; process[j].arrival_time <= process[i].burst_time+start_time && j<N; j++)
            if(process[i].arrival_time != process[i+1].arrival_time)
                sort(process+i+1, process+j, compareBurst);
            else
                sort(process+i, process+j, compareBurst);
        start_time += process[i].burst_time;
    }
}

```

Output

```

ubuntu@ubuntu:~/Documents$ g++ Prac8_3.c -o Prac8_3
ubuntu@ubuntu:~/Documents$ ./Prac8_3
Enter Number of Processes : 5

Enter Burst time of P[1] : 6
Enter Arrival time of P[1] : 2

Enter Burst time of P[2] : 2
Enter Arrival time of P[2] : 5

Enter Burst time of P[3] : 8
Enter Arrival time of P[3] : 1

Enter Burst time of P[4] : 3
Enter Arrival time of P[4] : 0

Enter Burst time of P[5] : 4
Enter Arrival time of P[5] : 4

PID Arrival time Burst time Waiting Time TAT
4      0           3       0     3
1      2           6       1     7
2      5           2       4     6
5      4           4       7    11
3      1           8      14    22

Total Waiting time : 26
Average Waiting time : 5.2

```

Que.	D. Priority Scheduling
Code	Non-Preemptive
	<pre>#include <iostream> #include <algorithm> using namespace std; typedef struct { int arrival_time, burst_time, waiting_time, pid, priority, TAT; } Process; bool compareArrival(Process a, Process b) { return (a.arrival_time < b.arrival_time); } bool comparePriority(Process a, Process b) { return (a.priority > b.priority); } void sortByPriority(Process process[], int N); int main() { int N, total_waiting_time = 0, start_time = 0; float average_waiting_time = 0; cout << "Enter Number of Processes : "; cin >> N; cout << endl; Process process[N]; for(int i=0; i<N; i++) { cout << "Enter Burst time of P[" << (i+1) << "] : "; cin >> process[i].burst_time; cout << "Enter Arrival time of P[" << (i+1) << "] : "; cin >> process[i].arrival_time; cout << "Enter Priority level of P[" << (i+1) << "] : "; cin >> process[i].priority; process[i].pid = (i+1); cout << endl; } sort(process, process+N, compareArrival); sortByPriority(process, N); cout << "PID Priority Arrival time Burst time Waiting Time TAT" << endl; start_time = process[0].arrival_time; for(int i=0; i<N; i++) { if(i == 0) process[i].waiting_time = 0; else process[i].waiting_time = start_time - process[i].arrival_time; total_waiting_time += process[i].waiting_time; start_time += process[i].burst_time; process[i].TAT = start_time - process[i].arrival_time; cout << process[i].pid << "\t" << process[i].priority << "\t" << process[i].arrival_time << "\t\t" << process[i].burst_time << "\t" << process[i].waiting_time << "\t\t" <<</pre>

```

process[i].TAT << endl;
}
average_waiting_time = (float)total_waiting_time/N;
cout << "\nTotal Waiting time : " << total_waiting_time << endl;
cout << "Average Waiting time : " << average_waiting_time << endl;
return 0;
}
void sortByPriority(Process process[], int N) {
int start_time = process[0].arrival_time;
int j;
for(int i=0; i<N-1; i++) {
for(j=i+1; process[j].arrival_time <= process[i].burst_time+start_time && j<N; j++);
if(process[i].arrival_time != process[i+1].arrival_time)
sort(process+i+1, process+j, comparePriority);
else
sort(process+i, process+j, comparePriority);
start_time += process[i].burst_time;
}
}

```

Preemptive

```

#include <iostream>
#include <algorithm>
using namespace std;
typedef struct {
int arrival_time, burst_time, waiting_time, pid, priority, TAT, remaining_time,
completion_time;
bool completed;
} Process;
bool compareArrival(Process a, Process b) {
return (a.arrival_time < b.arrival_time);
}
int main()
{
int N, total_waiting_time = 0, time = 0, completed = 0;
float average_waiting_time = 0;
cout << "Enter Number of Processes : ";
cin >> N;
cout << endl;
Process process[N];
for(int i=0; i<N; i++) {
cout << "Enter Burst time of P[" << (i+1) << "] : ";
cin >> process[i].burst_time;
cout << "Enter Arrival time of P[" << (i+1) << "] : ";
cin >> process[i].arrival_time;
cout << "Enter Priority level of P[" << (i+1) << "] : ";
cin >> process[i].priority;
}

```

```

process[i].completed = false;
process[i].remaining_time = process[i].burst_time;
process[i].pid = (i+1);
cout << endl;
}
sort(process, process+N, compareArrival);
time += process[0].arrival_time;
cout << "PID Priority Arrival time Burst time Waiting Time TAT Completion time" <<
endl;
while(completed < N) {
int max_priority = -1;
int id = -1;
for(int i=0; i<N; i++) {
if(process[i].completed == true) continue;
else if(process[i].arrival_time > time) break;
else {
if(process[i].priority > max_priority) {
max_priority = process[i].priority;
id = i;
}
}
}
time++;
process[id].remaining_time--;
if(process[id].remaining_time == 0) {
process[id].completed = true;
process[id].completion_time = time;
process[id].TAT = process[id].completion_time - process[id].arrival_time;
process[id].waiting_time = process[id].TAT - process[id].burst_time;
total_waiting_time += process[id].waiting_time;
completed++;
cout << process[id].pid << "\t" << process[id].priority << "\t" <<
process[id].arrival_time << "\t\t" << process[id].burst_time << "\t" <<
process[id].waiting_time << "\t" << process[id].TAT << "\t\t" <<
process[id].completion_time << endl;
}
}
average_waiting_time = (float)total_waiting_time/N;
cout << "\nTotal Waiting time : " << total_waiting_time << endl;
cout << "Average Waiting time : " << average_waiting_time << endl;
return 0;
}

```

Output

Non-Preemptive

```
ubuntu@ubuntu:~/Documents$ g++ Prac8_4_1.c -o Prac8_4_1
ubuntu@ubuntu:~/Documents$ ./Prac8_4_1
Enter Number of Processes : 5

Enter Burst time of P[1] : 4
Enter Arrival time of P[1] : 0
Enter Priority level of P[1] : 3

Enter Burst time of P[2] : 3
Enter Arrival time of P[2] : 0
Enter Priority level of P[2] : 2

Enter Burst time of P[3] : 7
Enter Arrival time of P[3] : 6
Enter Priority level of P[3] : 3

Enter Burst time of P[4] : 4
Enter Arrival time of P[4] : 11
Enter Priority level of P[4] : 1

Enter Burst time of P[5] : 2
Enter Arrival time of P[5] : 12
Enter Priority level of P[5] : 2
```

PID	Priority	Arrival time	Burst time	Waiting Time	TAT
1	3	0	4	0	4
2	2	0	3	4	7
3	3	6	7	1	8
5	2	12	2	2	4
4	1	11	4	5	9

Total Waiting time : 12
Average Waiting time : 2.4

Preemptive

```
ubuntu@ubuntu:~/Documents$ g++ Prac8_4_2.c -o Prac8_4_2
ubuntu@ubuntu:~/Documents$ ./Prac8_4_2
Enter Number of Processes : 5

Enter Burst time of P[1] : 4
Enter Arrival time of P[1] : 0
Enter Priority level of P[1] : 3

Enter Burst time of P[2] : 3
Enter Arrival time of P[2] : 0
Enter Priority level of P[2] : 2

Enter Burst time of P[3] : 7
Enter Arrival time of P[3] : 6
Enter Priority level of P[3] : 3

Enter Burst time of P[4] : 4
Enter Arrival time of P[4] : 11
Enter Priority level of P[4] : 1

Enter Burst time of P[5] : 2
Enter Arrival time of P[5] : 12
Enter Priority level of P[5] : 2
```

PID	Priority	Arrival time	Burst time	Waiting Time	TAT	Completion time
1	3	0	4	0	4	4
3	3	6	7	0	7	13
2	2	0	3	11	14	14
5	2	12	2	2	4	16
4	1	11	4	5	9	20

Total Waiting time : 18

Average Waiting time : 3.6

Practical 9

Aim: Find out the case where threads are helpful.

- A) With the help of posix API showcase the power of threads. Compare the Execution of single Process with threads execution.**
- B) Perform Thread synchronization using counting semaphores and mutual exclusion using mutex**

Case Where Threads Are Helpful:

Threads are useful when you want to perform multiple tasks concurrently, especially for applications that require parallel processing, such as:

- **Web Servers:** Handling multiple client requests simultaneously.
- **Scientific Computations:** Performing large computations concurrently to leverage multi-core systems.
- **Real-Time Systems:** Managing tasks in real time, like controlling sensors in IoT devices.
- **Multimedia Applications:** Processing multiple audio or video streams concurrently.

Threads are lightweight and can be more efficient than creating new processes because they share the same memory space and resources of the parent process.

Que.	A. With the help of posix API showcase the power of threads. Compare the Execution of single Process with threads execution.
Code	<p>1. Single Process Execution (Without Threads)</p> <pre>#include <stdio.h> #include <unistd.h> #include <time.h> void task1() { for (int i = 0; i < 5; i++) { printf("Task 1 - Iteration %d\n", i); sleep(1); } } void task2() { for (int i = 0; i < 5; i++) { printf("Task 2 - Iteration %d\n", i); sleep(1); } }</pre>

```

    }

int main() {
    clock_t start = clock();
    task1();
    task2();
    clock_t end = clock();

    double time_taken = ((double)(end - start)) / CLOCKS_PER_SEC;
    printf("Execution Time (Single Process): %.2f seconds\n", time_taken);

    return 0;
}

```

2. Multi-threaded Execution (Using POSIX Threads)

```

#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
#include <time.h>

void* task1(void* arg) {
    for (int i = 0; i < 5; i++) {
        printf("Task 1 (Thread) - Iteration %d\n", i);
        sleep(1);
    }
    return NULL;
}

void* task2(void* arg) {
    for (int i = 0; i < 5; i++) {
        printf("Task 2 (Thread) - Iteration %d\n", i);
        sleep(1);
    }
    return NULL;
}

int main() {
    pthread_t t1, t2;
    clock_t start = clock();

    pthread_create(&t1, NULL, task1, NULL);
    pthread_create(&t2, NULL, task2, NULL);

    pthread_join(t1, NULL);
    pthread_join(t2, NULL);

    clock_t end = clock();
}

```

```

        double time_taken = ((double)(end - start)) / CLOCKS_PER_SEC;
        printf("Execution Time (Multi-threaded): %.2f seconds\n", time_taken);

        return 0;
    }

```

Output

```

ubuntu@ubuntu:~/Documents/Practical9$ gedit Prac9_1_1.c
ubuntu@ubuntu:~/Documents/Practical9$ g++ Prac9_1_1.c -o Prac9_1_1
ubuntu@ubuntu:~/Documents/Practical9$ ./Prac9_1_1
Task 1 - Iteration 0
Task 1 - Iteration 1
Task 1 - Iteration 2
Task 1 - Iteration 3
Task 1 - Iteration 4
Task 2 - Iteration 0
Task 2 - Iteration 1
Task 2 - Iteration 2
Task 2 - Iteration 3
Task 2 - Iteration 4
Execution Time (Single Process): 0.00 seconds
ubuntu@ubuntu:~/Documents/Practical9$ gedit Prac9_1_2.c
ubuntu@ubuntu:~/Documents/Practical9$ g++ Prac9_1_2.c -o Prac9_1_2
ubuntu@ubuntu:~/Documents/Practical9$ ./Prac9_1_2
Task 1 (Thread) - Iteration 0
Task 2 (Thread) - Iteration 0
Task 1 (Thread) - Iteration 1
Task 2 (Thread) - Iteration 1
Task 1 (Thread) - Iteration 2
Task 2 (Thread) - Iteration 2
Task 1 (Thread) - Iteration 3
Task 2 (Thread) - Iteration 3
Task 2 (Thread) - Iteration 4
Task 1 (Thread) - Iteration 4
Execution Time (Multi-threaded): 0.00 seconds

```

Que.	B. Perform Thread synchronization using counting semaphores and mutual exclusion using mutex
Code	<p>1. Thread Synchronization with Semaphores:</p> <pre>#include <stdio.h> #include <pthread.h> #include <stdlib.h> #include <unistd.h> #include <stdint.h> #define NUM_THREADS 5 void* task(void* arg) { int thread_num = (intptr_t)arg; printf("Thread %d is running\n", thread_num); sleep(1); printf("Thread %d is done\n", thread_num); pthread_exit(NULL); } int main() { pthread_t threads[NUM_THREADS]; int rc; intptr_t t; for (t = 0; t < NUM_THREADS; t++) { printf("Creating thread %ld\n", t); rc = pthread_create(&threads[t], NULL, task, (void*)t); if (rc) { printf("ERROR; return code from pthread_create() is %d\n", rc); exit(-1); } } for (t = 0; t < NUM_THREADS; t++) { pthread_join(threads[t], NULL); } printf("All threads completed.\n"); return 0; }</pre>

2. Mutual Exclusion with Mutex:

```
#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>
#include <unistd.h>
#include <stdint.h>

#define NUM_THREADS 5

pthread_mutex_t mutex;

void* task(void* arg) {
    int thread_num = (intptr_t)arg;

    pthread_mutex_lock(&mutex);

    printf("Thread %d is running\n", thread_num);
    sleep(1);
    printf("Thread %d is done\n", thread_num);

    pthread_mutex_unlock(&mutex);

    pthread_exit(NULL);
}

int main() {
    pthread_t threads[NUM_THREADS];
    int rc;
    intptr_t t;

    pthread_mutex_init(&mutex, NULL);

    for (t = 0; t < NUM_THREADS; t++) {
        printf("Creating thread %ld\n", t);

        rc = pthread_create(&threads[t], NULL, task, (void*)t);
        if (rc) {
            printf("ERROR; return code from pthread_create() is %d\n", rc);
            exit(-1);
        }
    }

    for (t = 0; t < NUM_THREADS; t++) {
        pthread_join(threads[t], NULL);
    }

    pthread_mutex_destroy(&mutex);
}
```

	<pre> printf("All threads completed.\n"); return 0; } </pre>
Output	<p>1. Thread Synchronization with Semaphores:</p> <pre> ubuntu@ubuntu:~/Documents/Practical9\$ g++ Prac9_2_1.c -o Prac9_2_1 -lpthread ubuntu@ubuntu:~/Documents/Practical9\$./Prac9_2_1 Creating thread 0 Creating thread 1 Thread 0 is running Thread 1 is running Creating thread 2 Thread 2 is running Creating thread 3 Creating thread 4 Thread 4 is running Thread 3 is running Thread 0 is done Thread 1 is done Thread 2 is done Thread 3 is done Thread 4 is done All threads completed. </pre> <p>2. Mutual Exclusion with Mutex:</p> <pre> ubuntu@ubuntu:~/Documents/Practical9\$ g++ Prac9_2_2.c -o Prac9_2_2 -lpthread ubuntu@ubuntu:~/Documents/Practical9\$./Prac9_2_2 Creating thread 0 Creating thread 1 Creating thread 2 Creating thread 3 Thread 0 is running Creating thread 4 Thread 0 is done Thread 1 is running Thread 1 is done Thread 2 is running Thread 2 is done Thread 3 is running Thread 3 is done Thread 4 is running Thread 4 is done All threads completed. </pre>

Summary:

This practical exercise focused on implementing thread synchronization mechanisms using mutexes and semaphores in a multi-threaded environment.

Exercises Overview:

1. Power of Threads using POSIX API:

- Created multiple threads with `pthread_create()`.
- Compared single-process execution with multi-threaded execution, demonstrating improved performance due to parallelism.

2. Thread Synchronization with Mutexes:

- Implemented a critical section accessed by multiple threads.
- Used mutexes (pthread_mutex_t) to ensure mutual exclusion, preventing race conditions during access to shared resources.

3. Synchronization using Counting Semaphores:

- Demonstrated the use of semaphores (sem_t) to manage access to a limited resource among threads.
- Used functions like sem_init(), sem_wait(), and sem_post() to control concurrency and prevent resource contention.

Conclusion:

Practical 9 highlighted the efficiency of multi-threaded execution over single-process execution, emphasizing the role of threads in improving performance through concurrency. Synchronization techniques like mutexes and semaphores were essential in managing shared resources safely, ensuring mutual exclusion, and preventing race conditions. Threads are particularly beneficial in scenarios requiring concurrent execution, such as CPU-bound and I/O-bound tasks.

Practical 10

Aim: Implement inter process communication (IPC) using PIPEs and FIFOs.

Que.	Implement inter process communication (IPC) using PIPEs
Code	<pre>#include <stdio.h> #include <stdlib.h> #include <unistd.h> #include <string.h> int main() { int pipefds[2]; pid_t pid; char write_msg[] = "Hello from parent!"; char read_msg[50]; // Create pipe if (pipe(pipefds) == -1) { perror("pipe"); exit(EXIT_FAILURE); } pid = fork(); if (pid == -1) { perror("fork"); exit(EXIT_FAILURE); } if (pid > 0) { close(pipefds[0]); write(pipefds[1], write_msg, strlen(write_msg) + 1); close(pipefds[1]); } else { close(pipefds[1]); read(pipefds[0], read_msg, sizeof(read_msg)); printf("Child received: %s\n", read_msg); close(pipefds[0]); } }</pre>

	<pre> return 0; } </pre>
Output	<pre> ubuntu@ubuntu:~/Documents/Practical10\$ g++ Prac10_1.c -o Prac10_1 ubuntu@ubuntu:~/Documents/Practical10\$./Prac10_1 Child received: Hello from parent! </pre>

Que.	Implement inter process communication (IPC) using FIFOs.
Code	<p style="text-align: center;">Writer</p> <pre> #include <stdio.h> #include <stdlib.h> #include <string.h> #include <fcntl.h> #include <unistd.h> #define FIFO_FILE "/tmp/my_fifo" int main() { int fd; char write_msg[] = "Hello from FIFO writer!"; mknod(FIFO_FILE, 0666); fd = open(FIFO_FILE, O_WRONLY); write(fd, write_msg, strlen(write_msg) + 1); close(fd); return 0; } </pre> <p style="text-align: center;">Reader</p> <pre> #include <stdio.h> #include <stdlib.h> #include <fcntl.h> #include <unistd.h> #define FIFO_FILE "/tmp/my_fifo" int main() { int fd; char read_msg[50]; fd = open(FIFO_FILE, O_RDONLY); read(fd, read_msg, sizeof(read_msg)); printf("Reader received: %s\n", read_msg); close(fd); return 0; } </pre>

	}
Output	<pre>ubuntu@ubuntu:~/Documents/Practical10\$ g++ Prac10_Writer.c -o Prac10_Writer ubuntu@ubuntu:~/Documents/Practical10\$ g++ Prac10_Reader.c -o Prac10_Reader ubuntu@ubuntu:~/Documents/Practical10\$./Prac10_Writer ubuntu@ubuntu:~/Documents/Practical10\$./Prac10_Reader Reader received: Hello from FIFO writer!</pre>

Que.	The demonstration of fork () system call. Write a program which takes number of processes from the user and create those processes. (The fork system call will create processes in power of 2.)
Code	<pre>#include <stdio.h> #include <sys/types.h> #include <unistd.h> int main() { fork(); fork(); fork(); printf("Hello\n"); return 0; }</pre>
Output	<pre>ubuntu@ubuntu:~/Documents/Practical6\$ gcc -o Prac6_3 Prac6_3.c ubuntu@ubuntu:~/Documents/Practical6\$./Prac6_3 Hello Hello Hello ubuntu@ubuntu:~/Documents/Practical6\$ Hello Hello Hello Hello Hello Hello</pre>

Que.	The demonstration of fork () system call. Write a program which takes number of processes from the user and create those processes. (The fork system call will create processes in power of 2.)
Code	<pre>#include <stdio.h> #include <sys/types.h> #include <unistd.h> void forkexample() { if (fork() == 0) { printf("Child!\n");</pre>

	<pre> } else { printf("Parent!\n"); } } int main() { forkexample(); return 0; } </pre>
Output	<pre> ubuntu@ubuntu:~/Documents/Practical6\$ gcc -o Prac6_4 Prac6_4.c ubuntu@ubuntu:~/Documents/Practical6\$./Prac6_4 Parent! Child! </pre>

Summary:

This practical exercise focused on using Inter-Process Communication (IPC) with FIFO (named pipes) and unnamed pipes to enable communication and synchronization between processes.

Exercises Overview:

1. FIFO (Named Pipes):

- Created a FIFO using mkfifo() to enable communication between two unrelated processes: a writer and a reader.
- The writer sends a message through the FIFO, and the reader waits for and receives the message. This demonstrated how blocking I/O works to synchronize the processes.

2. Unnamed Pipes:

- Used the pipe() system call for communication between a parent and child process created with fork().
- The parent writes a message to the pipe, and the child reads it, showing how pipes can facilitate data exchange between related processes.

3. Synchronization with Threads:

- Demonstrated thread synchronization using mutexes and semaphores with POSIX threads.
- Threads shared a critical section of code, where mutexes ensured mutual exclusion, and semaphores regulated access to shared resources, preventing race conditions.

Conclusion:

Practical 10 showcased the use of IPC via FIFO and pipes for communication between processes and emphasized the role of thread synchronization using mutexes and semaphores to manage shared resources and avoid conflicts in multi-threaded environments.